Faculty of Computer Science and Information Technology

The Bacterial Foraging Optimisation Algorithm using Prototype Selection and Prototype Generation for Data Classification

Faizol Mohd Suria

Master of Science
2020

Bacterial Foraging Optimisation Algorithm using Prototype Selection and Prototype Generation for Data Classification

Faizol Mohd Suria

A thesis submitted

In fulfillment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2020

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

……

Signature

Name : Faizol Bin Mohd Suria

Matric No : 15020341

Faculty : Faculty of Computer Science and Information Technology,
Universiti Malaysia Sarawak

Dated : January 2020

ACKNOWLEDGEMENT

I would like to thank to Almighty Allah S.W.T for giving me strength and patience to complete my study. Then, I would like to express my gratitude to my supervisor Dr Mohammad Bin Hossin and my Co-Supervisor Dr Stephanie Chua Hui Li who always provided guidance, encouragement and support in my journey to complete this research. Without their persistent and commitment, I will not be able to finish this dissertation. Not to forget, I would like to thank all my friends in Postgraduate Laboratory for the constructive discussion and moral supports. I also want to thank G13 staffs who always helping me with tools and technical support to carry out my experiments.

I would like to thank my parent for their patience, financial and moral supports. Finally, I would like to acknowledge the support from Ministry of Higher Education for providing financial grant in this research.

ABSTRACT

A new emerging nature-inspired algorithm named Bacterial Foraging Optimisation Algorithm (BFOA) that mimics the foraging behaviour of E. coli bacteria has drawn lots of attention from other researchers due to its high convergence rate and global search capability compared to others. Technically, BFOA has been applied as supplementary algorithm for optimizing weight, parameters for other classifier algorithms and selecting optimised features for other classifiers. However, none of the available works had proposed BFOA as a classification algorithm despite of its good performance. Thus, this study aims to adopt and modify the BFOA into Instance Selection (IS) classifier by manipulating its global search capability and high convergence rate for data classification problem. There are two modified instance-based classifiers of BFOA were developed in this study. Both classifiers are inspired based on Prototype Selection (PS) and Prototype Generation (PG) known as BFOA-S and BFOA-G respectively. BFOA-G statistically outperformed BFOA-S by achieving 80.73% in the average testing accuracy with 5.16% average storage requirement, and 3 times faster in term of time complexity against BFOA-S in the benchmark experiment using 42 datasets. In addition, BFOA-G also performed well against ten existing IS algorithms by obtaining 83.1% in the average accuracy with 95.51% reduction rate and ranked first in the comparison study. On the other hand, BFOA-S showed competitive performance in the comparison against ten existing IS algorithms by obtaining 96.25% in reduction rate and ranked third in the ranking. Therefore, we conclude that the proposed BFOA-G is the best algorithm in the study, and PG approach is recommended for further development of BFOA as an IS algorithm.

Keywords: Bacterial foraging optimisation, data classification, instance-based classifier, nature-inspired algorithm

Algoritma Pengoptimuman Pengayaan Bakteria menggunakan Pemilihan Prototaip dan Penjanaan Prototaip untuk Pengkelasan Data

ABSTRAK

Algoritma Pengoptimuman Pengayaan Bakteria (APPB) yang menyerupai tingkah laku pencarian makanan oleh bakteria E. coli *telah menarik perhatian ramai penyelidik kerana mempunyai kadar penumpuan yang cepat dan keupayaan pencarian global lebih baik berbanding algoritma lain. APPB telah digunakan sebagai algorithma sampingan untuk mengoptimumkan berat, parameter untuk algoritma pengkelasan yang lain dan memilih ciri-ciri yang optimum untuk algoritma pengkelasan lain. Perkembangan terkini APPB menunjukkan bahawa APPB boleh diadaptasi dan digunakan sebagai algoritma pengklusteran untuk menyelesaikan masalah pengklusteran data. Walaubagaimana pun, setkat ini tiada lagi hasil kerja yang menggunakan APPB sebagai algoritma pengkelasan walaupun mempunyai reputasi yang baik dari segi keupayaannya. Oleh itu, kajian ini bertujuan untuk mengadaptasi dan mengubahsuai APPB sebagai algoritma Pemilihan Contoh (algoritma pengkelasan) bagi mengkaji keupayaan dan prestasi APPB dalam bidang pengkelasan data. Dua jenis pengkelas berasaskan contoh APPB telah dibangunkan dalam kajian ini. Kedua-dua pengkelas ini diilham daripada Pemilihan Prototaip dan Penjanaan Prototaip dan dikenali sebagai BFOA-S dan BFOA-G. BFOA-G secara statistik mengatasi keupayaan BFOA-S dengan mencapai 80.73% dalam purata ketepatan pengujian dengan purata keperluan penyimpanan sebanyak 5.16%, dan 3 kali lebih cepat dari segi kerumitan masa berbanding BFOA-S dalam eksperimen penanda aras dengan menggunakan 42 data. Tambahan lagi, BFOA-G juga berjaya menandingi sepuluh algoritma IS sedia ada dengan memperoleh 83.10% dalam purata ketepatan dengan kadar pengurangan 95.51% dan menduduki tempat pertama dalam kajian perbandingan. Di samping itu, BFOA-S*

menunjukkan prestasi yang kompetitif dalam perbandingan terhadap sepuluh algoritma IS sedia ada dengan memperoleh 96.25% dalam purata kadar pengurangan dan menduduki tempat ketiga dalam carta perbandingan. Oleh itu, BFOA-G adalah algoritma yang terbaik di dalam kajian ini dan kaedah PG disyorkan bagi pembangunan yang selanjutnya untuk APPB sebagai algoritma IS.

Kata kunci: *Pengoptimuman pengayaan bakteria, pengkelasan data, pengkelas berasaskan contoh, algoritma alam semula jadi*

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimisation
BFOA	Bacterial Foraging Optimisation Algorithm
GA	Genetic Algorithm
IS	Instance Selection
LS	Laplacian Score
LVQ	Learning Vector Quantization
PS	Prototype Selection
PG	Prototype Generation
PCA	Principal Component Analysis
PSO	Particle Swarm Optimisation
SOM	Self-Organizing Map

CHAPTER 1

INTRODUCTION

1.1 Overview

Nature-inspired algorithm is an interesting area of research that aims on designing and constructing innovative computing techniques by observing on how the nature behaves in various situations to solve complex problems. There are about 40 different nature-inspired algorithms have been proposed to solve many real-world optimisation problems (Fister et al., 2013). They are not only applicable to optimisation problem, but it can be extended to other domains with minimal modification.

Most of the nature-inspired algorithms that have been applied in other domains work as an optimisation algorithm to facilitate the process and improve another algorithm's performance. Besides that, the successfulness of the nature-inspired algorithms is not limited to improving other algorithms, but they also have been proposed as new algorithm in other domains. It applied on data mining problem such as data classification and data clustering that nature-inspired algorithms are preferable to be proposed as a new algorithm.

There are several examples of nature-inspired algorithms such as Genetic Algorithm (Kuncheva, 1997), Artificial Immune Systems (Carter, 2000), Particle Swarm Optimisation (Cervantes et al., 2007; Nanni & Lumini, 2009) and Ant Systems (Parpinelli et al., 2002) that have been proposed into new algorithm for data mining task. Their performances are comparable and most of them are better than non-nature-inspired algorithm. Recently, a new nature inspired algorithm named Bacterial Foraging Optimisation Algorithm (BFOA) has caught the attention of many researchers because of its successfulness in optimisation

domain compared to other nature-inspired algorithms (Hongwei & Liwei, 2015; Bensujin et al., 2016; Rani & Mangat, 2016; Lv et al., 2018).

BFOA was introduced in 2002 by Passino (2002) for solving optimisation problem. Since its inception, the BFOA has been applied successfully to engineering problems, such as optimal control (Kim & Cho, 2005), transmission loss reduction (Tripathy et al., 2006), timetabling (Shivakumar & Amudha, 2012) and image processing (Feng et al., 2012). According to survesy (Hongwei & Liwei, 2015; Bensujin et al., 2016; Rani & Mangat, 2016; Lv et al., 2018), BFOA has better optimisation capability compared to PSO and GA in term of global search capability. Furthermore, BFOA also have a high convergence rate that make the algorithm faster in finding optimal solution. In the recent development, BFOA has been actively being used in data analysis (Cho & Kim, 2011), as an optimiser in data classification (Damodaram & Phil, 2013) and already being introduced as new clustering algorithm (Lei et al., 2011; Garcia et al., 2012; Wan et al., 2012). In those works, it has been proven that BFOA has better performance as compared other successful algorithms such as GA, PSO and ACO. Although BFOA is actively being applied in data classification in a few studies, none of them has proposed BFOA as a new classification algorithm.

Basically, BFOA mimics the foraging behaviour of Escherichia coli (E.coli) bacteria, which exhibit chemotaxis, swarming, reproductive, elimination and dispersal process (Passino, 2002). Each of bacterium is representing the potential solution for an optimisation problem. An optimisation problem can be solved by minimizing or maximizing the objective function with the aim to find optimal values for that problem. In natural process, bacterium searches or forages for a better food source by swimming in rich nutrients concentration environment and tumble to avoids the noxious environment during chemotaxis process. The

decision-making process during the foraging is determined by two factors which are the concentration of the nutrients and the signals from the other foraging bacteria. Once a bacterium consumes enough nutrient the bacterium will undergo reproduction process with the presence of suitable temperature. During reproduction process, the bacterium will split in the middle and forms two identical individuals. In the presence of extreme condition, a group of bacteria either will be destroyed and dispersed into new environment in searching for rich food sources. This process is called elimination and dispersal.

As data grow bigger over the time, classification tasks become more challenging. The process to train data and produces a predictive model to classify the unknown input data become more difficult as the size of data grows. Most of the classical classifiers are unable to function properly and suffered several drawbacks due this challenge. Nearest Neighbour classification is an example of classifiers that critically suffered from several drawbacks due to the increasing size of the data. Nearest Neighbour Classification (kNN) is one the famous classification algorithm due to its simplicity and easy to implement, yet it has a powerful classification performance (Wu et al., 2008). However, it suffers several limitations such as high storage requirement, low noise tolerance and high computational time since it stores all the data to enable the classification capability.

Data reduction techniques are designed to overcome the limitation for classifiers that are unable to adapt the mass size of data. Reduction techniques can be categorized into feature selection, feature-value discretization and instance selection (Liu & Motoda, 2002). This study will emphasize Instance Selection as the reduction mechanism. There are some alternative names for Instance Selection which are reduction technique (Wilson & Martinez, 2000), Prototype Selection (Kruatrachue & Hongsamart, 2008; Garcia et al., 2012; Miloud-

aouidate et al., 2012; Triguero et al., 2012) and Prototype Reduction technique (Nanni & Lumini, 2011; Triguero et al., 2011; Triguero & Herrera, 2012). There are two main approaches in Instance Selection which are Prototype Selection (PS) and Prototype Generation (PG).

According to Garcia et al. (2012), PS and PG are considered as an optimisation problem since its objective is to find the optimal trade-off point between generalization ability, storage requirement and time complexity for Nearest Neighbour classification. Thus, it is preferable to adopt nature-inspired algorithms optimisation algorithm to solve PS and PG problems. Furthermore, the performance of other nature-inspired algorithms such as Genetic Algorithm, Ant Colony Optimisation and Particle Swarm Optimisation were better as compared to non nature-inspired algorithms as the PS or PG algorithm (Acampora et al., 2016). They have better searching mechanism and able to find the global solution. In the PS and PG term, those nature-inspired algorithms are commonly known as evolutionary algorithm since they use evolution of the populations to search for optimal solution (Cano et al., 2003; Garcia et al., 2010; Garcia et al., 2012; Triguero et al., 2012).

Since BFOA have advantage in optimisation capability, high convergence rate and global search capability compared to other successful nature-inspired optimisation algorithms, this study attempts to propose BFOA as Instance Selection classification algorithm which is focusing on producing an efficient algorithm with high generalization ability and small cardinality to improve the Nearest Neighbour Classification. To the best of our knowledge, the application of BFOA in and data classification still not thoroughly explored unlike PSO, GA and ACO. Thus, it is a great opportunity to study the full potential of BFOA in another domain.

1.2 Problem Statements

Recently, a new nature-inspired optimisation algorithm named Bacterial Foraging Optimisation Algorithm(BFOA) has emerged and shown better performance than other successful nature-inspired algorithms such as GA, ACO and PSO in term of global search capability and convergence rates (Hongwei & Liwei, 2015). Many of the nature-inspired algorithms have successfully been adopted as the classification algorithm especially as an Instance Selection (IS) algorithm. The IS algorithm is an instance-based classifier with aims to overcome the limitation of Nearest Neighbour Classification. The problem with Instance Selection is it difficult to search for the optimal trade-off between reduction rate and generalization performance without degrading the accuracy of the reference set or model. This problem is considered as NP-problem since the solutions only can be obtained through exhaustive search in large problem space. Furthermore, there is no guarantee that the obtained solution is the optimal solution for the problem (Ho et al., 2002).

Good optimisation algorithms such as Genetic Algorithm (Kuncheva, 1997), Artificial Immune Systems (Carter, 2000), Particle Swarm Optimisation (Cervantes et al., 2009; Nanni & Lumini, 2009) and Ant Systems (Parpinelli et al., 2002) become favourable choices to be adopted as an IS classification algorithm because they have better capability and faster in searching for solutions compared to exhaustive search algorithms.

The conversion of those optimisation algorithms into IS algorithms are different to each other due to the differences in the structure of the algorithms. There are many new IS algorithms have been introduced with the aim to produce an efficient IS algorithm with good generalization capability to solve nearest neighbour classification. In the latest surveys, PSO is currently a popular optimisation algorithm for the development of IS algorithm (Garcia et

al., 2012; Triguero et al., 2012; Sakinah & Ahmad, 2014; Rosales-Pérez et al., 2017). However, it is questionable either BFOA can be adopted as new IS algorithms or not since there is no works done before in adopting BFOA as a classification algorithm.

The problem with the conversion of BFOA into IS algorithm is that original structure of the algorithm cannot be converted directly into IS. The main reason is the current representation of solution and information encoding in original BFOA are not suitable and it is impossible to adopt the IS concept directly into the algorithm's structure. Furthermore, there are two major approaches in IS that are different in their concept. Both approaches need different kind of representation and the modification only can be made based on the solution representation (Cano et al., 2003; Derrac et al., 2010). Without suitable modification on the algorithm's structure, the algorithm is unable to process the information to produce the desired results. Since there is no work done in BFOA for IS before, it is unclear which solution representation is suitable and what kind of modification can be made to enable the conversion of BFOA into IS.

1.3 Research Questions

From the problem that has been mentioned, there are some questions that are needed to be pointed out. The research questions that will be answered are described as follows.

i. How to design an optimisation BFOA into IS algorithms that based on PS approach and PG approach?
ii. Which approaches between PS and PG is the best classifier for BFOA as an IS algorithm?

iii. How is the performance of the proposed BFOA algorithms as an IS algorithm as compared to other IS algorithms based on generalization, reduction rate and time efficiency?

1.4 Research Objectives

There are four specific objectives to be fulfilled in this research. The objectives are as follows:

i. to design a data classification version of BFOA using prototype selection and prototype generation approaches;
ii. to analyse both proposed algorithms with existing IS algorithms (nature-inspired and non-nature-inspired) based on generalization ability (accuracy), reduction rate and time efficiency using several data sets (vary in terms of domain, dimensionality and volume);
iii. to recommend the best modified version of BFOA for data classification problem.

1.5 Research Scope

This study focuses on the development and conversion of an optimisation algorithm named Bacterial Foraging Optimisation Algorithm (BFOA) into Instance Selection (IS) algorithm to improve the IS technique in finding the optimal subset of representative of a dataset for nearest neighbour classification. Since there is no work done on BFOA as a classifier before, thus this research will design and develop two IS algorithm based on Prototype Selection (PS) and Prototype Generation (PG) approaches to adopt the optimisation BFOA into IS classifiers. The performance of the proposed algorithms will be evaluated using 42 various domains of benchmark datasets from the UCI machine learning

and compared with ten existing IS algorithms. The datasets used in the experiments consist of small and medium size only. Majority of the datasets used were imbalanced, but it will not be discussed and covered in this study.

1.6 Research Contributions

There are two major contributions that have been achieved in this study. They are listed as follows.

Contribution 1: The study has proposed a framework for the conversion of BFOA into IS algorithm that based on PS approach to improve nearest neighbour classification. The proposed framework was successfully implemented and produce an IS algorithm named as Prototype Selection Bacterial Foraging Optimisation Algorithm or in short as BFOA-S. This framework was used as a pioneer study for the latter improvement of BFOA with PS approach.

Contribution 2: A framework has been proposed for the conversion of BFOA into IS algorithm that based on PG approach to improve nearest neighbour classification. The proposed framework was successfully developed and named as Prototype Generation Bacterial Foraging Optimisation Algorithm or in short as BFOA-G. This study was used as a pioneer study for further development of BFOA with PG approach.

1.7 Organisation of the Thesis

There are six important chapters presented in this thesis. A brief explanation of each chapter is described as follows:

Chapter 1 is introducing a brief explanation on the research problems encounter in this research field, research objectives, research scopes and research contribution.

Chapter 2 presents the literature review on the background of BFIA and its existing application in data mining. This chapter continues with the review on Instance Selection and its major categories and present in details on the previous works that related to the conversion or modification of nature-inspired algorithm for Instance Selection.

Chapter 3 presents the research methodology used to conduct the study. All dataset involved and preparation method for these datasets are explained in this chapter. This chapter also present the evaluation methods that have been used to evaluate the performance of the proposed algorithms.

Chapter 4 presents the detailed explanation on the conversion framework for both proposed algorithms. This chapter also presents the parameter configuration for the proposed algorithm and discuss their effect on the performance of the algorithms.

Chapter 5 presents the experimental results and the analysis using statistical tests to evaluate the performance of the proposed algorithms. An additional comparison against existing IS algorithms are explained and discussed in this chapter. Overall finding is discussed to highlight important finding to improve the proposed algorithms.

Finally, Chapter 6 draws conclusion of this study. Limitation and future work are also presented in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter presents the background and recent works on the nature-inspired algorithm named Bacterial Foraging Optimisation Algorithm (BFOA). The objective of this thorough review is to study the basic concepts, characteristics, structures, capabilities and the successfulness of BFOA application in various domains. The timeline of BFOA's application is presented to describe the current trends on the development of BFOA in data mining and especially focusing on data classification. Apart from that, this chapter also includes the literature on the modification of other nature-inspired optimisation algorithms into Instance Selection (IS) algorithms to study how the modification is being made for IS. Two approaches in the IS that were reviewed in this study which is Prototype Selection and Prototype Generation. This review shows the general idea on how the other optimisation algorithm can be converted into an IS algorithm. It has been used as a reference for the modification of an optimisation BFOA into a data classification algorithm, mainly as an IS algorithm.

2.2 Bacterial Foraging Optimisation Algorithm

Bacterial Foraging Optimisation algorithm (BFOA) is an optimisation algorithm proposed by Passino (2002) to solve optimisation problem and control system. The details of on biological background and the application of BFOA in optimisation approach are explained in next subsections.

2.2.1 Biological Background

BFOA works by mimicking the foraging behaviour of Escherichia coli (E. coli) bacteria that refers to the foraging strategy of bacteria swarms in multi-optional function optimisation. Foraging strategy is a method for locating, handling and ingesting food by animals. Bacteria search and obtain nutrients in a manner to maximise the energy intake. By sending signals, it enables an individual bacterium to communicate with other bacteria.

Biologically, an E. coli bacterium has a plasma membrane, cell wall, and capsule that contains the cytoplasm and nucleoid. It has several flagella that randomly distributed around its cell wall as shown in Figure 2.1. The flagella play an important role for the bacterium locomotion or so-called chemotaxis. They move in a rotation manner and counter clockwise direction that enable the bacteria to move forward with large displacement. This process called “swim”. When the flagella move clockwise, bacteria will move in a random direction with very small displacement and it is called “tumble” process. With the help from flagella, a bacterium able swim in a very fast pace due to its tiny in size. The size of E. Coli bacterium is about 1 μm in diameter, 2 μm in length, 1 picogram in weight with about 70% of it being water.

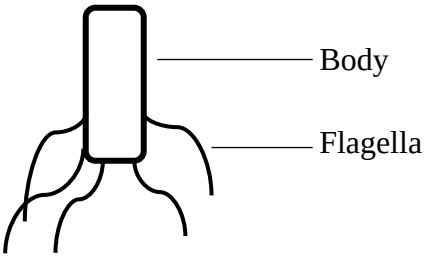


Figure 2.1: A Bacterium Cell

E. Coli bacteria always try to search for a place that contain higher concentration of nutrients and try to avoid any noxious places using a specific locomotion pattern called “taxes”. While climbing up to high concentration of nutrient area, bacteria will release a chemical substance called attractant and they release repellent substance when countered harmful places. A bacterium swims continuously in similar direction and it will spend more time in a high concentration environment. This pattern keeps repeating as it counters a favourable environment. Otherwise, the movement pattern will change if it encounters a lower concentration of nutrient (noxious environment). The bacterium will move in random direction as an escape mechanism to avoid harmful place and this process is called “tumble” process. These foraging behaviour processes are occurred in a major process known as “chemotaxis”. Figure 2.2 shows how the tumble and swim process are done. The chemotaxis process can be carried out due to the sensing capability that reside in the bacterium.

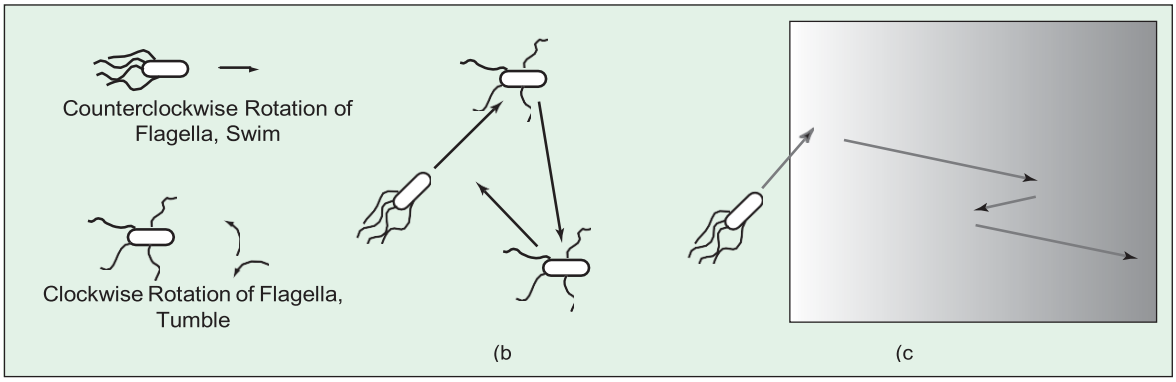


Figure 2.2: Locomotion of E. Coli Bacteria (Passino, 2002)

As mentioned earlier, bacteria have demonstrated an interesting group behaviour when they are placed in a semisolid nutrient media. They form an intricate stable spatiotemporal pattern (swarms) of bacteria and travel by moving up the nutrient gradient. This event occurred because the bacteria cells are stimulated by a high level of *succinate*

(repellent) that triggered them to release an attractant *aspartate*. The attractant helps the bacteria to aggregate into groups and resulting a high bacterial density that move together to a better environment. This pattern is formed based on cell-to-cell signal and foraging stimuli.

Once a bacterium consumes enough nutrients (a healthy bacterium), it will result to the significant increase in the bacterium's length and size. Then, the bacterium will split in the middle to form an exact replica of itself in the presence of suitable temperature. On other hand, the unhealthy bacterium (bacterium with poor foraging capability) will die or eliminate from the population. This process is called reproduction. The mechanism of replication of healthy bacteria will keep the population constant and it ensures only bacteria with a good foraging strategy are able to survive for the next cycle. Under some circumstances, the population of bacteria will change due to some external factors. For example, a sudden significant increase in temperature will cause the death of bacteria population. It is not only limited to the death of bacteria population, but a group of bacteria can disperse into a new environment with some trigger from the external factors.

2.2.2 Bacteria Foraging Optimisation Algorithm Concept

The biological concept of the foraging E. coli is modelled into an optimisation technique by adopting their foraging mechanism. Suppose that a problem needs to find the global minimum of $J(\theta), \theta \in p$, where θ is the position of the bacterium and $J(\theta)$ represents the nutrient level in the medium at the current θ. There are no measurements or an analytical description of the gradient $\nabla J(\theta)$. Here, it can be seen as a minimization problem which it can be solved using non-gradient optimisation technique adopted from the foraging behaviour of E. Coli bacteria. There are three possible condition of $J(\theta)$ which are $J(\theta) < 0$, $J(\theta) = 0$, and $J(\theta) > 0$. These values indicate the nutrient level at the current position θ of a

bacterium either rich, neutral or noxious. Then, the foraging of bacteria will occur in the manner of climbing up the nutrient concentration ($J(\theta) < 0$), avoiding the noxious environment ($J(\theta) > 0$) and search for the way out when the environment is neutral.

The basic optimisation concept of BFOA consists four principal mechanisms known as chemotaxis, swarming, reproduction, and elimination-dispersal. There are several important parameters need to be considered before applying BFOA to solve a problem. The parameters are listed and described in the Table 2.1.

Table 2.1: List of Parameters in Bacterial Foraging Optimisation Algorithm

Symbol	Meaning
P	Dimension of the search space
S	Total number of bacteria in the population
N_c	The number of chemotactic steps
N_s	The swimming length
N_{re}	The number of reproduction steps
N_{ed}	The number of elimination-dispersal events
P_{ed}	Elimination-dispersal probability
$C(i)$	The size of the steps taken in the random direction specified by the tumble

2.2.2.1 Chemotaxis

Chemotaxis is a process that simulates the movement of a bacterium searching for food. There are two types of movements that reside in chemotaxis which are swimming and tumbling. A bacterium moves in these two modes alternately in its entire lifetime with the help of bacterium's flagella during foraging process (Das et al., 2009). Bacteria swim for a longer period or distance in a high concentration of nutrients (friendly environment) and avoid a noxious environment by tumbling.

In the optimisation perspective, chemotaxis is a process perform by the bacteria in a manner of minimizing the cost function (lower *J(θ)* value). Let denotes the position of each bacterium in the population with *P(j, k, l) =*{ $\theta^i(j,k,l)$| *i =1,2,..., S}* with its corresponding nutrient value at *j*-th chemotactic step, *k*-th reproduction step, and *l*-th elimination-dispersal events. Then, *J(i, j, k , l)* is denoted as the cost at the location of the *i*-th bacterium. The cost is referring to the concentration of the nutrient and it will be the important criterion to determine the direction of movement of bacteria in the next iteration. In a formal mathematical notation, chemotaxis is represented in Equation 2.1,

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{(\Delta T(i)\Delta(i))}} \qquad \text{Equation 2.1}$$

where *C(i)* is denoted the length of steps taken in the random direction manner specified by the tumble during the run and *Δ (i)* is a random vector with each element lying in [−1, 1]. If the value of $J(i+1,j,k,l)$ at position $\theta^i(j+1,k,l)$ is lower than *J(i, j, k , l)* at previous position, then the bacterium will move in a similar direction into the position $J(i+1,j,k,l)$ with the predefined *C(i).* This process is repeated until the necessary condition is met. The movement pattern is called swim. The bacterium will stop to swim when the maximum value of N_s is met. Then, the bacterium needs to tumble to forage.

2.2.2.2 Swarming

Swarming process is occurred along the foraging process of the bacteria. The bacteria release an attractant when they consume enough succinate due to chemical reaction in the cell. This attractant signals other bacteria as a trigger to keep them closer to each other and

congregate them into groups. Once they receive the signal, they will move as concentric pattern of groups with high density of bacteria climbing the nutrient concentration. As the bacteria has the capability to release attractant, they also can release repellent to avoid other bacteria from consuming its nearby nutrients. It is not possible to have two bacteria at the same nutrient location since the resource is limited. The swarming process is represented in Equation 2.2,

$$J_{cc}\left(\theta, P(j,k,l)\right) = \sum_{i=1}^{S} J_{cc}^{i}\left(\theta, \theta^{i}(j,k,l)\right)$$

$$= \sum_{i=1}^{S}\left[-d_{attract}\, exp\left(-w_{attract}\sum_{m=1}^{P}\left(\theta_m - \theta_m^i\right)^2\right)\right] +$$

$$\sum_{i=1}^{S}\left[-h_{repellent}\, exp\left(-w_{repellent}\sum_{m=1}^{P}\left(\theta_m - \theta_m^i\right)^2\right)\right] \quad \text{Equation 2.2}$$

Which denotes the combination of cell-to-cell attraction and repellent affect, where $J_{cc}(\theta, P(j, k, l))$ is the objective function value that to be added to the actual objective function to present time varying objective function. The notation of S is denoting the total number of bacteria while p is denoting the number of variables involved in the search space. On the other hand, $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^T$ is representing a point on the optimisation domain, and θ^i_m is denoting the m-th components of the ith bacterium position θ^i. d_{attract}, w_{attract}, $h_{\text{repellant}}$, and $w_{\text{repellant}}$ are different coefficients used for signalling. Some researchers claim that swarming works as a penalty function to the actual objective value. The idea is to let the algorithm to converge into an optimal value. When the penalty value becomes zero, it is expected that the bacteria are converged in an optimal value.

2.2.2.3 Reproduction

Reproduction is a process of eliminating poor bacteria and reproducing the good bacteria to maintain the population bacteria with strong survival capability. This event can be seen clearly when the bacteria population increase significantly in a concentration nutrient media and decrease rapidly when there is low concentration of nutrient present in the medium. When a bacterium consumes enough nutrient, its length and size will increase. Then, the bacterium is ready to reproduce. For the optimisation technique, reproduction process is started after maximum N_c chemotactic size is met. The cost function of each bacterium in the population is used as indicator to determine their health. The lower cost function value indicates healthier bacterium and higher cost value means that the bacterium does not consume enough nutrient (poor health). In a formal notation, the health of the bacteria is represented in Equation 2.3.

$$J_{health} = \sum_{j=1}^{Nc+1} J(i,j,k,l) \qquad \text{Equation 2.3}$$

The bacteria in a population is sorted in ascending order based on the accumulative of their cost function for each bacterium. Then, they will be divided into two halves, S_r which is represented in Equation 2.4. The first half is the bacteria with a better health and the other half is the poor health bacteria.

$$S_r = \frac{S}{2} \qquad \text{Equation 2.4}$$

The poor health bacteria will die and the bacteria with better health will reproduce. A bacterium will split into two individuals and both have similar characteristics with their mother at the same location. This mechanism maintains the size of bacteria in the population. Once the bacteria are reproduced, they will undergo chemotactic process again and repeat the whole process until the maximum iteration of N_{re} is met.

2.2.2.4 Elimination and Dispersal

The bacteria population is easily affected by the external factors such as sudden rises of temperature or catastrophic that can cause the bacteria to be eliminated or dispersed into a new environment. This event is called the elimination and dispersal. It plays an important role in preventing algorithm from trapped at the local solution. In the algorithm, this event is started after maximum iteration of N_{re} is met. To model the elimination and dispersal process, each bacterium is supplied with a probability between 0 to 1. Through the value of the probability, it will determine whether it will be eliminated and dispersed. A predefined threshold, P_{ed} is used to indicate which bacterium will eliminated and reinitialized into new region. Bacterium with a lower probability value than P_{ed} is eliminated and dispersed by reinitialized it into new region and bacterium with higher probability value than threshold P_{ed} will not affected. After the elimination and dispersal process complete, bacteria will undergo chemotaxis and followed by reproduction again. This process is repeated until maximum iteration of N_{ed} is met. The detailed pseudocode for BFOA can be referred in Figure 2.3.

[Step 1] Initialize parameters $p, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, C(i)(i=1,2\ldots S)$.

[Step 2] Elimination-dispersal loop: $l=l+1$

[Step 3] Reproduction loop: $k=k+1$

[Step 4] Chemotaxis loop: $j=j+1$

a) For i =1,2…S take a chemotactic step for bacterium i as follows.

b) Compute fitness function, J(i, j, k, l).
Let, $J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j, k, l), P(j, k, l))$ (i.e. add on the cell-to-cell
attractant–repellant profile to simulate the swarming behaviour) where, Jcc is defined in (2).

c) Let $J_{last}=J(i, j, k, l)$ to save this value since a better cost could be found via a run.

d) Tumble: generate a random vector $\Delta(i) \in \Re p$ with each element i, m =1,2,..., p, a Δ random number on [-1, 1].

e) Move: Let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{(\Delta T(i)\Delta(i))}}$$

This results in a step of size C (i) in the direction of the tumble for bacterium i.

f) Compute $J(i, j+1, k, l)$ and let
$J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l))$

g) Swim

i. Let $m = 0$ (counter for swim length).

ii. While $m < N_s$ (if have not climbed down too long).

- Let $m=m+1$.
- If J (i, j + 1, k , l) < Jlast (if doing better), let $Jlast = J(i, j + 1, k, l)$ and let

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{(\Delta T(i)\Delta(i))}}$$

And use this $\theta^i(j+1,k,l)$ to compute the new J (i, j + 1, k , l) as we did in (f)

- Else, let m= N_s . This is the end of the while statement.

h) Go to next bacterium (i+1) if $i \neq S$ (i.e., go to (b) to process the next bacterium).

[Step 5] If $j < N_c$, go to step 4. In this case, continue the chemotaxis process since the life of the bacteria is not over.

[Step 6] Reproduction:

a) For the given k and l, and for each i = 1,2,..., S , let

$$J_{health}^{i} = \sum_{j+1}^{N_c+1} J(i,j,k,l)$$

Be the health of bacteria i (a measure of how many nutrients it got over its lifetime
and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic
parameters $C(i)$ in order of ascending cost J_{health} (higher cost means lower health).

b) The S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values will split (this process is performed by the copies that are made are placed at the same location as their parent).

[Step 7] If $k < N_{re}$, go to step 3. In this case, the algorithm had not reached the number of specified reproduction step, so it starts the next generation of the chemotactic loop.

[Step 8] Elimination-dispersal: For i = 1,2..., S with probability P_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimisation domain. If $l < N_{ed}$, then go to step 2; otherwise end.

Figure 2.3: Pseudocode for Bacterial Foraging Optimisation Algorithm (Passino, 2002)

2.3 Bacterial Foraging Optimisation Algorithm in Data Mining

Originally, Bacterial Foraging Optimisation Algorithm (BFOA) was invented by Passino (2002) for control system. Since its inception, BFOA has been applied successfully to engineering problems, such as optimal control (Kim & Cho, 2005), transmission loss reduction (Tripathy et al., 2006), timetabling (Shivakumar & Amudha, 2012), image processing (Feng et al., 2012) and harmonic estimation (Mishra, 2005). To the best of our knowledge, the use of BFOA in data analysis (Cho & Kim, 2011) and data classification (Damodaram & Valarmathi, 2013) are not thoroughly explored by the researchers. There a

lot of potential areas that BFOA can be applied to improve the current performance. The latest development of BFOA, it has been employed on data clustering to improve the clustering performance (Olesen et al., 2009; Lei et al., 2011; Wan et al., 2012). The application of BFOA in several areas are shown as below.

2.3.1 Classification

This study has categorized the current implementation of BFOA for data classification into two categories which are optimising the feature selection and optimizing the parameters of the existing classifiers. This categorisation is based on the role of BFOA in the previous works.

The role of BFOA in feature selection is to determine directly or indirectly (combined with other feature selection algorithm) the most relevant features and discard the noise features to prepare the data for further classification process. Most of the BFOA current applications in feature selection are in medical domain (Jakhar et al., 2011; Rani & Mangat, 2013; Bensujin et al., 2016; Rani et al., 2016; Sindhu, 2016). Especially for the image data where the dimension of the features is large, and it is a necessity to employ feature selection to speed up the process. Feature selection works by selecting the most relevant features and remove the redundant one. By reducing the redundant and noise features, it will improve the classification results.

The implementation of BFOA in those previous works are motivated by the successfulness of other evolutionary algorithms such as GA, ACO and PSO in improving the current selection method. Since BFOA has advantage in optimisation especially in term of global search capability as compared to other evolutionary algorithms, it draws the researchers' attention to implement it for feature selection as summarized in Table 2.2.

BFOA has been implemented in various ways for feature selection. Some researchers solely use BFOA to select relevant features and some of them combine BFOA with other evolutionary algorithms such as PSO to improve the performance. PSO is currently a popular algorithm that has been applied in many applications. However, due to some limitation in PSO, BFOA is used to overcome the issue (Olesen et al., 2009; Zhang & Wu, 2012; Wan et al., 2012; Chen et al., 2014). BFOA shows a better result in selecting the relevant features compared to the other optimisation algorithms such as Particle Swarm Optimisation (PSO) (Jakhar et al., 2011; Rani & Mangat, 2016), Principal Component Analysis (PCA) and Laplacian Score (LS) which are popular algorithms for feature selection. Furthermore, the combination of PCA and LS is unable to outperform the performance of BFOA for feature selection (Rani & Mangat, 2016).

Besides, it is found that BFOA is also being used to optimise the learning capability of classifiers during the training process. There are several ways to enhance learning performance of classifiers such as optimising weight and optimising the parameters of the classifier. Most of algorithms behaviours were controlled by their parameters to halt the process or to force some conditions during the process. The enhancement of the learning capability of classifier is done through finding the optimal parameters or weights for the classifier. The convergence rate of the algorithm is depended on these parameters that also affects the product of the result. However, the process of optimising the parameters of an algorithm is considered optimisation problem because there is no clear relation between the parameters to the performance of the algorithm. Each parameter in the algorithm has its own role and it behaves differently to each other. Hence, it is not an easy task to predict the optimal values for these parameters. The recommendation to overcome this issue is to

employ the stochastic optimisation algorithms to predict the optimal values for the parameters.

Table 2.2: Application of BFOA in Data Classification

Data classification problems	**Authors**	**Role of BFOA**
Noise features present in dataset	Jakhar et al., (2011)	BFOA was used to further reduce the extracted features of Cambridge Online Research Laboratory (ORL) gray-scale face dataset that made by Discrete Cosine Transform (DCT) technique.
	Rani & Mangat, (2013)	BFOA was adapted to select relevant features in the Pima Indian Diabetes dataset to improve the generalization ability of Feed-Forward Back Propagation neural network classifier.
	Kora & Kalva, (2015)	A hybrid BFOA and PSO known as BFPSO has been employed for feature selection of ECG signals and these selected features have been used as the input for Levenberg-Marquadt Neural network classifier.
	Bensujin et al., (2016)	The frequency dependent adaptive chemotactic bacterial foraging optimisation algorithm (FDABFOA) has been proposed to optimise and fine-tune the extracted features of ST elevation Myocardial Infraction (STEMI) dataset
	Rani et al., (2016)	BFOA was utilized at the pre-processing stage for feature selection in ANN classifier to discard irrelevant features in lung image dataset for the ANN and SVM classifiers.
	Sindhu, (2016)	BFOA was employed to select a small subset of informative or relevant genes from thousands of genes of cancer expression dataset.
Low learning capability of classifier	Chakrabarty et al., (2012)	Employed BFOA to optimise the kernel function of Support Vector Machine (SVM) for hyperspectral image classification
	Qiang & Ai-Min, (2013)	BFOA has been applied to optimise the operation parameters of SVM regression model.
	Putra et al., (2014)	The BFOA has been employed to optimise the weights and bias parameter of Backpropagation Neural network to predict Forex Gold Index (XAUUSD)
	Chen et al., (2014)	The hybrid PSO and BFOA was introduced known as BFPSO to optimise Neural Fuzzy Classifier (NFC) for several benchmark datasets and skin colour detection problem.

Table 2.2 continued

Low learning capability of classifier	Al-Hadi et al., (2011)	BFOA was applied to optimise the weights and parameters of Feed-Forward Neural Network for several benchmark datasets.
	Varghese et al., (2012)	BFOA was used to tune the Backpropagation Neural network parameters for MRI images of Alzheimer Disease dataset
	Chakrabarty et al., (2012)	Employed BFOA to optimise the kernel function of Support Vector Machine (SVM) for hyperspectral image classification
	Qiang & Ai-Min, (2013)	BFOA has been applied to optimise the operation parameters of SVM regression model.
	Putra et al., (2014)	The BFOA has been employed to optimise the weights and bias parameter of Backpropagation Neural network to predict Forex Gold Index (XAUUSD)
	Chen et al., (2014)	The hybrid PSO and BFOA was introduced known as BFPSO to optimise Neural Fuzzy Classifier (NFC) for several benchmark datasets and skin colour detection problem.
	Kaur & Kaur, (2014)	BFOA has been applied to adjust the weight and parameter values of Feed-Forward Neural Network and Cascade-Forward Neural Network
	Pal et al., (2014)	Hybrid BFOA and Learning Automata algorithm have been employed to identify the optimal features subset from a given imagery electroencephalography (EEG) based on brain-computer interfacing (BCI) dataset
	Lv et al., (2018)	BFOA has been improved by adopting opposition-based learning strategy and it becomes new algorithm called IBFO for parameter optimisation in kernel extreme learning machine (KELM). The study is proposed to improve the prediction of severity of somatization disorder.

2.3.2 Clustering

The involvement of BFOA is not limited in classification task only, it is also has been applied in clustering for pattern discovery. Clustering task can be defined as a task to discover the classes by itself through partitioning the training data into clusters based on the similarity of attributes (features). In literature, clustering task is known as an unsupervised learning (Greene et al., 2008).

The earliest work on BFOA for data clustering was on 2009 (Olesen et al., 2009). The authors proposed two algorithms which are a new clustering algorithm based on BFOA named AutoCB and a hybrid approach between PSO and BFOA as a new clustering algorithm called AutoPCB. Since then, many works have been proposed to study the capability of BFOA to solve the data clustering problem as shown in Table 2.3. A study has proposed a new clustering algorithm based on BFOA using different fitness functions to improve clustering results (Wan et al., 2012). The results show that BFOA outperformed the popular k-means algorithm, PSO-based and ACO-based clustering algorithm in term of clustering performance and it manages to handle data sets with various cluster sizes, densities and multiple dimensions. Furthermore, a study was conducted to improve a BFOA-based clustering algorithm by incorporating the cooperative approach (Hongwei & Liwei, 2015). The internal structures of the algorithm were modified to make it more stable and accurate. This study reported that the proposed algorithm able to distinguish the sample precisely and automatically improving the clustering quality.

Apart from that, Zhang and Wu (2012) has proposed a new fitness function called Variance Ratio Criterion (VRC) to find the maximum value of the proposed fitness function to improve cluster analysis (Zhang & Wu, 2012). This cluster analysis needs to be converted into an optimisation to be able to apply the proposed method. It is reported that BFOA is a better optimiser for cluster analysis as compared to GA and combinatorial PSO (CPSO) since it can find the highest value with a least time (Zhang & Wu, 2012). It concludes that BFOA is an efficient algorithm. Instead of being proposed as a new algorithm, BFOA's advantage has also been exploited to overcome the influence of cluster numbers on the experimental result on Protein-Protein Interaction (PPI) networks (Lei et al., 2011). Based on result reported, BFOA can improve the clustering result. It is proven that BFOA does not rely on

the initial point to obtain the optimal solution. It is possible because BFOA is flexible during the searching process thus it can move freely in the search space. Thus, the initial points do not affect the performance of algorithm. In addition, a comparative study has been carried out to evaluate the performance of BFOA in clustering against two famous evolutionary optimisation algorithms PSO and GA (Rupal & Kataria, 2014). The result reported that BFOA outperformed PSO and GA by having higher value of Cohesion, Precision, Recall and small value of Variance.

Table 2.3: Application of BFOA for Data Clustering

Data clustering problem	**Authors**	**Role of BFOA**
Difficulty in finding optimal cluster to represent clusters	Olesen et al., (2009)	A novel clustering algorithm using BFOA which is known as AutoCB. Then, there is an improve version of AutoCB with PSO named as AutoPCB to sharpen the chemotactic steps in BFOA.
	Lei et al., (2011)	BFOA was used as a base algorithm to overcome the influence of cluster number on experimental result of clustering Protein to Protein Interaction (PPI) networks.
	Wan et al., (2012)	A new clustering algorithm based on BFOA which is known as BF-C to find the center or representation for each cluster by optimizing the objective function. The Sum Squared Error (SSE) was used as a fitness function to evaluate the representativeness.
	Zhang & Wu, (2012)	Adopted the BF-C with new fitness function (variance ratio criterion) to determine the optimal cluster points.
	Rupal & Kataria, (2014)	BFOA is used to enhance the accuracy H-Mean clustering. It is a comparative study to evaluate the performance of three evolutionary optimisation algorithms which are BFOA, GA and PSO.
	Hongwei & Liwei, (2015)	A cooperative approach of BFOA for clustering, which is known as Cooperative Bacterial Foraging Optimisation (CBFO) to improve clustering results.

2.3.3 Regression

Regression is a task in data mining that make prediction of numerical value by building a model based on one or more predictors. Table 2.4 shows the existing works on

BFOA for regression problem. BFOA was integrated with regression models to improve its prediction and performances. The implementation of BFOA in regression started in 2012 where it has been used to improve the robot routing by optimizing the interpolation of linear regression to make robots moving in a curving manner and remove conventional zigzag motion (Darvishi et al., 2012). Since then, a few works have been proposed such as adopting BFOA as an optimisation tool to reduce error in the models for ground water prediction (Kumar, 2014) and as a parameter optimisation tool for LS-SVM regression model for the Burning-Through-Point (BTP) forecasting (Song et al., 2016). In summary, BFOA is a flexible algorithm that can be applied in many domains. However, full capability of BFOA in regression is not fully explored yet. For instance, it can be adopted as a BFOA-based regression algorithm which is similar to the clustering domain.

Table 2.4: Application of BFOA for Regression

Regression problems	**Authors**	**Role of BFOA**
Errors in regression models	Darvishi et al., 2012	BFOA was used to improve the robot improve the robot routing by optimizing the interpolation of linear regression to make robots moving in a curving manner and remove conventional zigzag motion.
	Kumar, 2014	BFOA was used as an optimisation tool for linear regression and Multiple Linear Regression to reduce errors in the models.
	Song et al., 2016	BFOA was adapted to optimise the parameters of LS-SVM regression model for the Burning-Through-Point (BTP) forecasting.

2.4 Instance Selection

Instance Selection is one of the data reduction methods by scaling down data. There are three types of data reduction method which are feature selection, feature-value discretization and instance selection (Liu & Motoda, 2002; García-Pedrajas et al., 2010). Nowadays, instance selection is also known with other alternative names such as Prototype

Selection (Kruatrachue & Hongsamart, 2008; Garcia et al., 2012; Miloud-aouidate, Baba-ali, Alia, & Ezzouar, 2012; Triguero et al., 2012), Prototype Reduction technique (Nanni & Lumini, 2011) and reduction technique (Wilson & Martinez, 2000). In instance selection, its main objective is to reduce the number of rows in the dataset while maintaining the original goal of a data mining process as if the whole dataset is being used. By minimizing the size of the data, it is possible to reduce the computational cost of the data mining algorithm and improve the generalization capability since less noise is presented.

The Nearest Neighbour classification is quite powerful classifier instead of its simplicity, it suffered some critical limitations. Those limitations are large storage requirement of prototypes, long response time classification and have low noise tolerance (Kuncheva & Bezdek, 1998; Liu & Motoda, 2002; Garcia et al., 2012; Triguero et al., 2012). NN classifier takes longer time (time complexity) and very sensitive to noise when it processes a larger size of dataset because it needs to process the whole dataset to enable the classification task. Thus, it critically depended on the quality of the data to have a better generalization capability. Those limitation can be solved by applying instance selection method since instance selection objective is to reduce the number of the dataset. As mentioned before, instance selection pursues reduction, generalization ability (classification performance) and time complexity (Garcia et al., 2012; Triguero et al., 2012) simultaneously which is reducing the size of dataset will remove the noises present in the dataset and simultaneously reduces the time complexity of the classifier.

Technically, IS method is specifically designed to overcome the NN classifier's limitation. However, there are some challenges faced by instance selection that it is always struggles to balance between the optimal size of subset of the representatives and the mining

quality. Reducing the number of instances sometime lead to the loss of information and results in reducing the mining quality. Due to this trade-off, instance selection is considered as an optimisation problem (Liu & Motoda, 2002).

Since instance selection can be seen as an optimisation problem, the optimisation technique like stochastic heuristic search can be used to locate the optimal subset of representatives (a set of the prototype (reference set) or combination of instances) in a search space that can balance between the size of subset of representatives and mining quality. The earliest work on adopting optimisation technique for instance selection was carried out by Kuncheva and Bezdek (1998). The study investigates the effectiveness of heuristic and stochastic technique compared to the random search and clustering-based method. Genetic Algorithm (a nature-inspired optimisation algorithm) was converted into an instance-based classifier (using 1-NN rule) and was compared with other searching strategies. It turned out that heuristic and stochastic method able to find a better subset of representatives compared to the other searching strategies. Since then, several works were proposed on modifying the optimisation algorithms into the instance selection algorithms in order to find which optimisation framework is the best in finding the optimal subset of representatives.

Instance selection consists of two major approaches which are prototype selection and prototype generation (abstraction) (Garcia et al., 2012; Triguero et al., 2012). Algorithms that developed with these approaches can be categorised with another subcategory under each approach. They are evolutionary and non-evolutionary IS algorithms. The evolutionary term was introduced by Cano et al. (2003) to address the algorithm that process and produce result based on the evolution of population. The details explanation for both approaches and subcategories will be discussed in following subsections.

2.4.1 Prototype Selection Approach

In general, Prototype Selection (PS) approach is a method of selection of subset of relevant prototypes(instances) from the original dataset that can performs similar or better as if the whole dataset is used. In a formal notation of PS problem, let $S \subseteq TR$ become the subset of a training set named as *TR*, and if there are query instances that need to be classified named as *X*, then classify them using the subset *S* instead of using the whole training set where the classification accuracy using subset *S* must be similar or better than using the whole data in *TR*. Figure 2.4 portrays how the selection mechanism work.

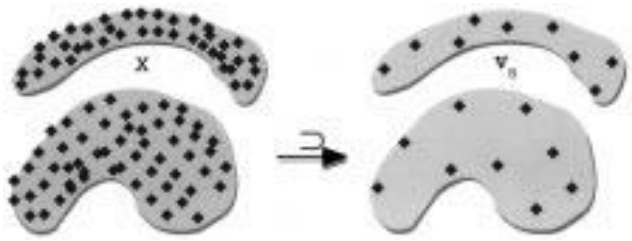


Figure 2.4: Prototype Selection in Instance Selection (Kuncheva & Bezdek, 1998)

The main advantage of PS approach is its capability to select relevant prototypes without generating new prototypes to represent the whole dataset. PS approach is suitable for the data that have been managed and each of classes has their own prototypes. In many real-world applications, data are collected and managed well. Thus, generating new data to replace them are irrelevant because sometimes the generated data could not represent the real situation in some problem or domain.

According to the study conducted by Garcia et al. (2012), there are more than 50 available PS algorithms that have been proposed to improve the nearest neighbour classification problem. PS algorithms can be categorised into evolutionary PS algorithm and non-evolutionary PS algorithm. Evolutionary PS algorithms are derived from the nature-

inspired algorithms such as Genetic Algorithm (GA) (Kuncheva & Bezdek, 1998; Bezdek & Kuncheva, 2001; Gil-Pita & Yao, 2007; Suguna & Tanushkodi, 2010), Evolutionary Algorithm (EA) (Cano et al., 2005; Garcia-Pedrajas et al., 2010), Ant Colony Optimisation (ACO) (Miloud-Aouidate & Baba-Ali, 2012; Miloud-Aouidate & Baba-Ali, 2013), and Particle Swarm Optimisation (PSO) (Hu & Tan, 2015). All these algorithms preserved its originality and applied stochastic heuristic search to locate the solution (a reference set) by transforming this searching problem into optimisation problem. For non-evolutionary PS algorithms, they were developed without the evolutionary concept. The examples of successful non-evolutionary PS algorithm are IB3 (Aha et al.,1991), DROP3 (Wilson & Martinez, 2000), ICF (Brighton & Mellish, 2002) and there are a lot of non-evolutionary PS algorithms that have been proposed until today.

This study focuses on the development of evolutionary PS algorithms. The evolutionary PS algorithms work directly with the classification performance as their objective function and optimise them using evolutionary computation to get the optimal prototypes. Through evolutionary strategy, the algorithm searches for global best solution through the selection of the best combination of prototypes. Along the process, these prototypes can be evolved, eliminated and reproduced to obtain better objective function. The earliest work on evolutionary PS algorithm was proposed by Kuncheva and Bezdek (1998) by adopting Genetic Algorithm (an evolutionary algorithm) into a PS classifier. Since then, researchers started to take interest in other evolutionary algorithms which are mostly based on natured-inspired algorithm. The following reviews are the recent development of evolutionary PS algorithm for nearest neighbour classification.

Gil-Pita and Yao (2007) adopted genetic algorithm for editing k-nearest neighbour classifier. It aims to improve the classification performance of the classifier by proposing three techniques such as using objective function called mean square error, the implementation of a clustered crossover and a fast-smart mutation scheme. New clustered crossover minimized the objective function by considering the relationship between the different bits of the bit stream. On the other hand, fast-smart mutation scheme helps the algorithm to select the best gene by taking account the variations of objective function for a given set of each gene. It managed to improve the classification accuracy of k-NN classifier compared to standard k-NN.

Kruatrachue and Hongsamart (2008) proposed a combination of MCSI method and Genetic Algorithm (GA) for nearest neighbour classification. This study focuses on the condensing the prototypes to find the minimal consistent subset of prototypes. The condensation problem is considered as hard combinatorial problem because there is no exact solution can be obtained in a limited time for the problem. The weakness of MCSI is it always trapped at local minimum and this disadvantage was tackled by GA. GA is used for global search and MCS1 is responsible to find the local solution. By exploiting both advantages, the proposed algorithm has significant difference in the cardinality of the prototypes for small dataset compared to standard GA and MC1. However, the difference in performance is not so significant for the large dataset (more than 700 instances).

Garcia et al. (2008) proposed a model of memetic algorithm that incorporates an ad hoc local search for optimizing the prototype selection problem named Steady-State Memetic Algorithm (SSMA). Memetic algorithm is an evolutionary algorithm that shares similar concept to the popular evolutionary algorithm such as genetic algorithm that follow

the survival of the fittest through evolution. The result shows that the proposed algorithm outperformed the non-evolutionary algorithm and had a competitive result against the other evolutionary prototype selection algorithms in term of accuracy and reduction rate.

In other work, García-Pedrajas et al. (2010) proposed cooperative coevolutionary instance selection (CCIS) for instance-based learning. This study used divide and conquer concept by decomposing a complex problem into simpler subproblems. This method is called as cooperative coevolution technique. The simpler problems are being solved separately and the result will be combined to obtain the global solution. The instances are divided using stratification approach. However, stratification process will make the algorithm suffers from the partial knowledge it has on the dataset. In term performance, the proposed algorithm shows a good performance in term of generalization ability and reduction rate as compared to the other evolutionary algorithm such as genetic based instance selection algorithm.

Figueredo et al. (2012) proposed an IS algorithm based on natural immune system's suppression mechanism. The author claimed that the proposed algorithm was faster and demand less computational resources than the evolutionary method. It focuses on condensing the training instances to reduce redundancies in dataset and this training set can be applied on any classifier. However, in their proposed method, the author employed the nearest neighbour rule to determine the smallest size of the relevant training set. The proposed algorithm achieves better classification accuracy compared to CHC (genetic algorithm based) algorithm. However, CHC have better reduction rate as compared to the proposed method.

Miloud-Aouidate & Baba-Ali (2012) proposed a hybridization of ant colony optimisation (ACO) algorithm with K-nearest neighbour. The authors focus on the condensation technique for prototype selection approach. They aim for the minimal size of consistent subset from the training set that can improve the standard nearest neighbour classification by exploiting the ACO optimisation capability. The proposed algorithm named ANT-KNN consist of three steps which are the creation of the associate's chain, the application of condensing technique and noise filter process to remove the prototypes that cause misclassification. The reference prototypes obtained from the training will be used for classification using 1-NN rule. ANT-KNN outperforms standard KNN, TRKNN and DROP3 in term of reduction rate and classification accuracy.

Sakinah & Ahmad (2014) proposed a method using cooperative binary PSO for nearest neighbor classification. The authors adopt feature selection and instance selection in their works. It aims for the relevant prototypes and select the most relevant features to improve the classification performance. The proposed method divides the candidate solution into several sub-components named sub-swarms. The proposed method only considered the accuracy performance and does not include the reduction rate in the evaluation.

2.4.2 Prototype Generation Approach

Prototype Generation (PG) is a suitable method to solve the problems had by NN classifier especially when it comes to the data that do not have instances to represent some classes. Although PG has an almost similar problem and goals with Prototype Selection (PS) approach, but they are slightly different compared to each other. PS approach is using the original training data to find the optimal subset of representative instances through the process of discarding the noise and redundant instances. However, PG approach can

generate and replace the original instances with new artificial instances which is contradicted to PS approach (Garcia et al., 2012; Triguero et al.,2012). However, both PG and PS are sharing the same goals that they want to reduce the training data as much as possible and simultaneously attempt to achieve the highest possible classification accuracy using 1-nearest neighbour (1-NN) when dealing with the unseen data (Garcia et al., 2010).

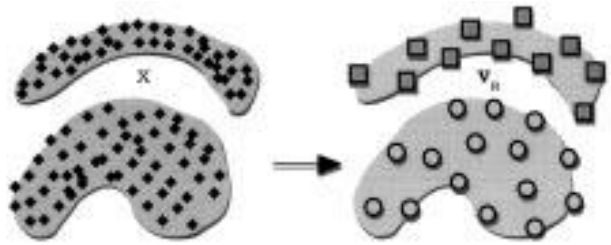


Figure 2.5: Prototype Generation in Instance Selection (Kuncheva & Bezdek, 1998)

Similar to PS approach, researchers started to improve PG algorithms by adopting evolutionary algorithm to improve its performance over NN classification. The empirical study shows that the evolutionary PS algorithm outperformed the non-evolutionary PS algorithms (Nanni & Lumini, 2009). As a result, abundance of evolutionary PS algorithms were produced by adopting nature-inspired algorithm such as genetic algorithm (GA) (Kuncheva & Bezdek, 1998; Bezdek & Kuncheva, 2004; Gil-Pita, 2007; Suguna & Tanushkodi, 2010), evolutionary algorithm (EA) (Cano et al., 2005; Garcia-Pedrajas et al., 2010), ant colony optimisation (ACO) (Miloud-Aouidate & Baba-Ali, 2012; Miloud-Aouidate & Baba-Ali, 2013) and Particle Swarm Optimisation (PSO) (Chang, 1974; Cervantes, 2007; Triguero & Garcia, 2011; Hu & Tan, 2015). All these algorithms preserve its originality and applied stochastic heuristic search to locate the solution (a reference set) by transforming this searching problem into optimisation problem.

Over the years, a lot of algorithms have been proposed using prototype generation method. Prototype generation can be categorised into three groups which are data condensation approach, prototype tuning approach and evolution-based approach (Hu & Tan, 2015). The general idea for condensation-based prototype generation algorithms is it works by condensing the data without taking NN classification into consideration. It assumes the prototype generation as a common condensation problem. The examples of condensation based prototype generation algorithms are vector quantization (Xie et al., 1993), self-organizing map (SOM) (Kohonen, 1990; Li, Manry et al., 2005) and Gaussian mixture model (Lozano et al., 2006). Most of these algorithms usually performed poorly in NN classification as compared to other prototype generation algorithms that had been incorporated with some strategies that includes the NN classification performance.

The prototype tuning approach involves with relabelling, merging, moving or deleting the training instances to generate prototypes. This approach is usually followed by heuristic mechanisms to ensure good performance in NN classification. However, there are some studies that solely use merge operation based on instance's class to generate prototypes (Chang, 1974) and some of them adopted different method such as bootstrap to merge training instances to produce prototypes (Hamamoto, Uchimura, & Tomita, 1997). In addition, prototype tuning also can be carried out by editing scheme (Koplowitz & Brown, 1981), moving instances (Geva & Sitte, 1991) where the human-defined rules are introduced to enable the movement of prototypes and converting the prototype generation problem into optimisation problem by tuning its cost function (Decaestecker, 1997) to obtain better classification performance. In brief, the prototype tuning approach depends on the human-defined rules to tune the prototype and establish correlation toward NN classification performance. These rules are expected to improve the classification performances. However,

it is not guaranteed that the classification performance will be optimal due to the difficulty in establishing clear relationship between rules and final classification performance.

Since most of the nature-inspired algorithms are converted into evolutionary IS algorithm, thus this study will focus more on the evolution-based prototype generation approach. This approach is recognized as performance-driven approach (Hu & Tan, 2015). The evolution-based prototype generation algorithms work directly with the classification performance as their objective function and optimise them using evolutionary computation to obtain the optimal prototypes or reference set. The algorithm will search for the global best solution through the evolution of prototypes that during the process, the prototypes can be altered, eliminated and reproduced to obtain a better objective function. There are many existed PG algorithms and their numbers are steadily increasing until today.

Cervantes et al. (2009) in their proposed PG algorithm was based on PSO with Michigan approach called AMPSO. The authors encoded each particle as a prototype and assigned each of them with a fixed class. In Michigan approach, an individual of the population does not encode the whole solution to the problem but only a part of it. Thus, the whole swarm of particles in AMPSO represented the potential solution to the problem. Due to the representation of particles, local fitness was introduced to evaluate prototypes. This approach was able to reduce the dimension of the search space since each particle was encoded as a single prototype (instance) and it offered a flexible size of prototypes in the population (represented the solution to a problem). Besides, AMPSO also allows insertion and removal of new particles in swarm to produce a good combination of prototypes.

Nanni and Lumini (2009) proposed prototype generation algorithm based on PSO. The study generates prototypes called particles by selecting k numbers of instances randomly

from training data and encodes them in form of vectors. Each particle represents a potential solution to a problem. The proposed algorithm works by minimizing the error rate of the training process of each particle through PSO mechanism. Due to randomness in generating particles, the operation is repeated several times to produce high performance particle. Voting rule is used for the classification process of unknown instances.

Triguero et al. (2011) proposed a hybrid of prototype selection and position adjustment with Differential Evolution (DE) or PSO. The prototypes are encoded using random stratified selection of prototype that have similar ratio to the number of instances in each class. This method ensures at least one instance is encoded into an individual. It means that one individual is having many instances that represents a potential solution to a problem.

In 2014, the development of evolutionary Prototype Generation algorithm has extended using the multiobjective method. Rosales-Pérez et al. (2014) introduced Multiobjective (MO) scheme for evolutionary Prototype Generation algorithm called EMOPG and as they consider the Prototype Generation problem as multiobjective optimisation problem. The objective of Prototype Generation is to find the optimal solution that can produce higher classification performance with small cardinality of solution (high reduction rate). The author stated that the normal PG algorithm only focussed on maximizing the classification performance. As suggested, the algorithm with MO scheme attempt to optimise two objectives separately by employing Pareto optimality to find the optimal trade-off between classification performance and reduction rate. In MO case, there is no single solution that can optimise and satisfy both conflicting objectives. At the end of the process, MO returns a set of solutions instead of a single solution. A weighting scheme is proposed to measure the instance's discrimination power in the training set. Furthermore, the weight

is also being used to generate the initial prototypes. Position adjustment using PAES (Pareto Archived Evolution Strategy) is employed to move the prototypes in order in order to maximize the classification performance and simultaneously reduce the cardinality of the solution. PAES will return a set of non-dominated solutions and Tchebycheff metric is used to select a single solution from them. Later, EMOPG has been extended by the authors by adopting feature selection to maximise the classification accuracy (Rosales-Pérez et al., 2017).

Hu and Tan (2015) introduced new evolution-based prototype generation algorithm called MOPSO. The authors encoded the prototypes based on the distribution of data in each class with prespecified reduction rate to ensure each there is at least one instance for each class in a single prototype or particle. A particle in the swarm is represented a potential solution for a problem. Then, PSO mechanism is deployed to optimise the objective function to search for good solutions. Multiobjective (MO) and error rank are deployed as the fitness function for the proposed algorithm to improve classification performance. Error rank is proposed specifically to improve the generalization ability of NN classifier by taking the misclassified instances into consideration for evaluation.

2.5 General Framework of An Evolutionary IS Algorithm

As explained in formal notations in previous section, evolutionary PS and PG share some similarities in general framework for their implementation. Figure 2.6 simplified both formal notations into a figure to show the basic idea for implementing the evolutionary PS and PG algorithms.

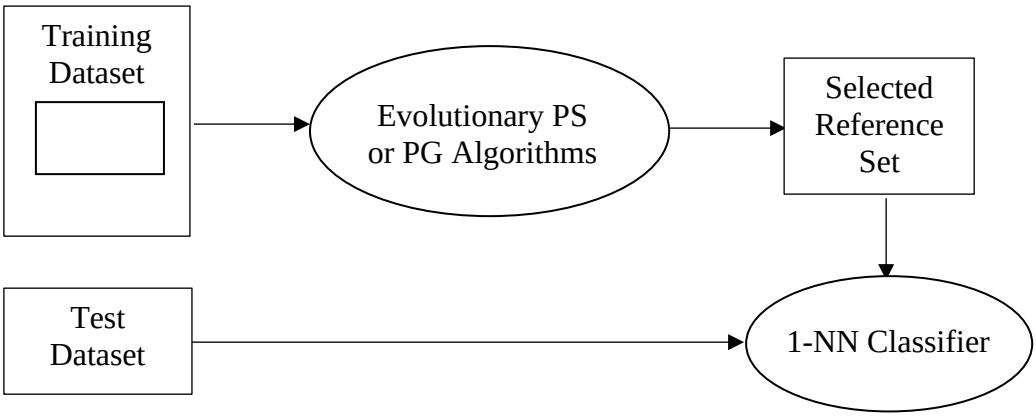


Figure 2.6: Basic Steps for Evolutionary Prototype Selection and Prototype Generation Processes

The evolutionary IS algorithms process the training set by selecting or generating a subset of instances that can represent the whole dataset. After several iterations or cycles, a small cardinality of reference set (selected or generated) will be produced as a model for the problem. The classification of the unknown instances will be using the reference set instead of the whole dataset to assign class to them by using 1NN rule. This is the basic procedure of an evolutionary IS algorithm processes.

There are two important criteria that need to be considered in developing evolutionary IS algorithm that PS and PG both approaches shared. They are the representation for the population and fitness function that will evaluate the performance of the candidate solutions (Cano et al., 2003; Derrac et al., 2010). The detail for each criterion is discussed as follow.

2.5.1 Data Representation

Representation of the data involved with encoding information of instances into a form that can be process by evolutionary PS or PG algorithms to produce solution for a problem. This is one of the important criteria need to be considered before developing the

evolutionary PS and PG algorithms. According to Cano et al. (2005), the representation of solution (prototype) can also affect the efficiency of the evolutionary IS. Improper representation will not able to bring out the full potential of the algorithm.

Generally, evolutionary algorithm represent individual in a population in form chromosome, thus the individual in evolutionary IS algorithms is normally represented in a string-like chromosome form that contain the information about the dataset (Cano et al., 2005; Garcia-pedrajas et al., 2010). Every individual (known as prototype) undergoes evolution process within specific cycles and evolves into a better solution for a problem at hand. There are two ways to represent solution in an individual or prototype (some researchers used encoding term) for evolutionary IS algorithm. They are Pittburgh and Michigan approaches (Cervantes et al., 2005). Pittburgh approach encoded a solution into a single individual in a population and a population is comprised with many individuals. This approach is commonly used in most of the standard evolutionary IS algorithms. On the contrary, Michigan approach encoded a part of solution into a single individual of the population. Thus, the whole population is representing a solution for a problem. From the literatures, most of discussion on these approaches are towards the evolutionary PG algorithms and evolutionary PS algorithm did not employed Michigan approach for solution encoding. According to Cervantes et al. (2005, 2007, 2009) in his comparison work, Michigan approach is more efficient compared to Pittburgh on PSO for evolutionary PG algorithms.

Once the solution representation is decided, the next step is to encode the information into individuals in the population. The typical information encoding scheme for evolutionary IS algorithm is the binary representation. In binary encoding scheme, every instance will be

assigned with value either ‘1’ or ‘0’ that indicate selected instances or not selected instances in an individual. Value of ‘1’ indicates the selected instances and ‘0’ indicates non-selected instances. Thus, an individual in population will be represented in a string (chromosome) with combination of ‘0’ and ‘1’ as depicted in Figure 2.7. Only selected instances will be evaluated to produce the solution for the problem. This method was employed by the earliest work on evolutionary IS algorithm by Kuncheva (1995). There is a continuation on the usage of this encoding scheme in latter proposal for evolutionary IS algorithms (Ho et al., 2002; Cano et al., 2005; García et al., 2008; Kruatrachue & Hongsamart, 2008; Ahmad, 2014). From the trend, most of evolutionary PS algorithms used binary encoding more compared to evolutionary PG algorithms.

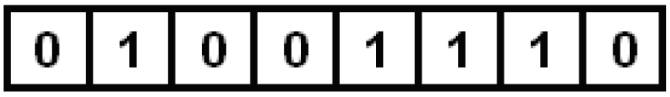


Figure 2.7: Binary Representation Scheme

There are other information encoding available such as real number (Suguna & Thanushkodi, 2010; Miloud-Aouidate & Baba-Ali, 2012; Figueredo et al., 2012), integer number and other alternative encoding schemes (Rosales-Pérez et al., 2016). The real number encoding is used to encode the position index of instances in dataset into the prototypes or encode continuous values such as normalized value of the attributes since they are normally ranged from [0,1].

2.5.2 Fitness Function

Most of the evolutionary IS algorithms employed fitness function to evaluate the quality of the solution. Based on the problem at hand, the fitness function is set either to minimize or maximize the value of the fitness. Most of IS algorithms are naturally employed

the classification accuracy (becomes reference set) to measure the generalization ability of the classifier or solution. On the other hand, reduction rate of the size of selected instances from the original size of dataset is also commonly used to measure the performance of evolutionary IS algorithms (Derrac et al., 2010; Garcia et al., 2012; Triguero et al., 2012). Most of evolutionary PS considered both of accuracy and reduction rate performance in determining better solution.

On the other hand, most of PG only employed classification accuracy obtained from the generated reference sets or solution (classification with 1NN). Interestingly, the reduction rate is not considered as the performance measure since generally the size of solution generated by PG is normally small (Derrac et al., 2010). They are usually in a fixed concrete value. Thus, the size of solution did not give much impact on efficiency of the algorithm and it can focus solely on maximising the quality of the generated solution.

2.6 Summary

This chapter has discussed two important components for the study. The first component in this chapter has explained the BFOA's background, structure, capability and its performance in its original domain which is optimisation. Then, a thorough reviews have been done on the application of BFOA in data mining to study the current trend of the development for BFOA. The second component is this study focusses on data classification especially in Instance Selection. This chapter has included the background of the Instance Selection and its advantages for the data classification. Finally, this chapter present the current development and modification on the nature-inspired algorithms using Prototype Selection and Prototype Generation approaches as IS algorithms.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview

The objective of this chapter is to present the methodology used in conducting the study. This chapter discusses all the research steps involved. Besides, it also describes in detail all the benchmark datasets that are being used in the experimental study. Furthermore, this chapter is also explained in detail on pre-processing the benchmark datasets to prepare them for experiment. All system requirements, evaluation and analysis methods that are being used in this study are discussed in detail. Finally, this chapter is ended with a summary that concludes all materials in this chapter.

3.2 Research Flow

The processes and procedures used in this study will be elaborated in this section. This study employs the combination of build and empirical research methodology. The summarisation of the research procedures is shown in Figure 3.1.

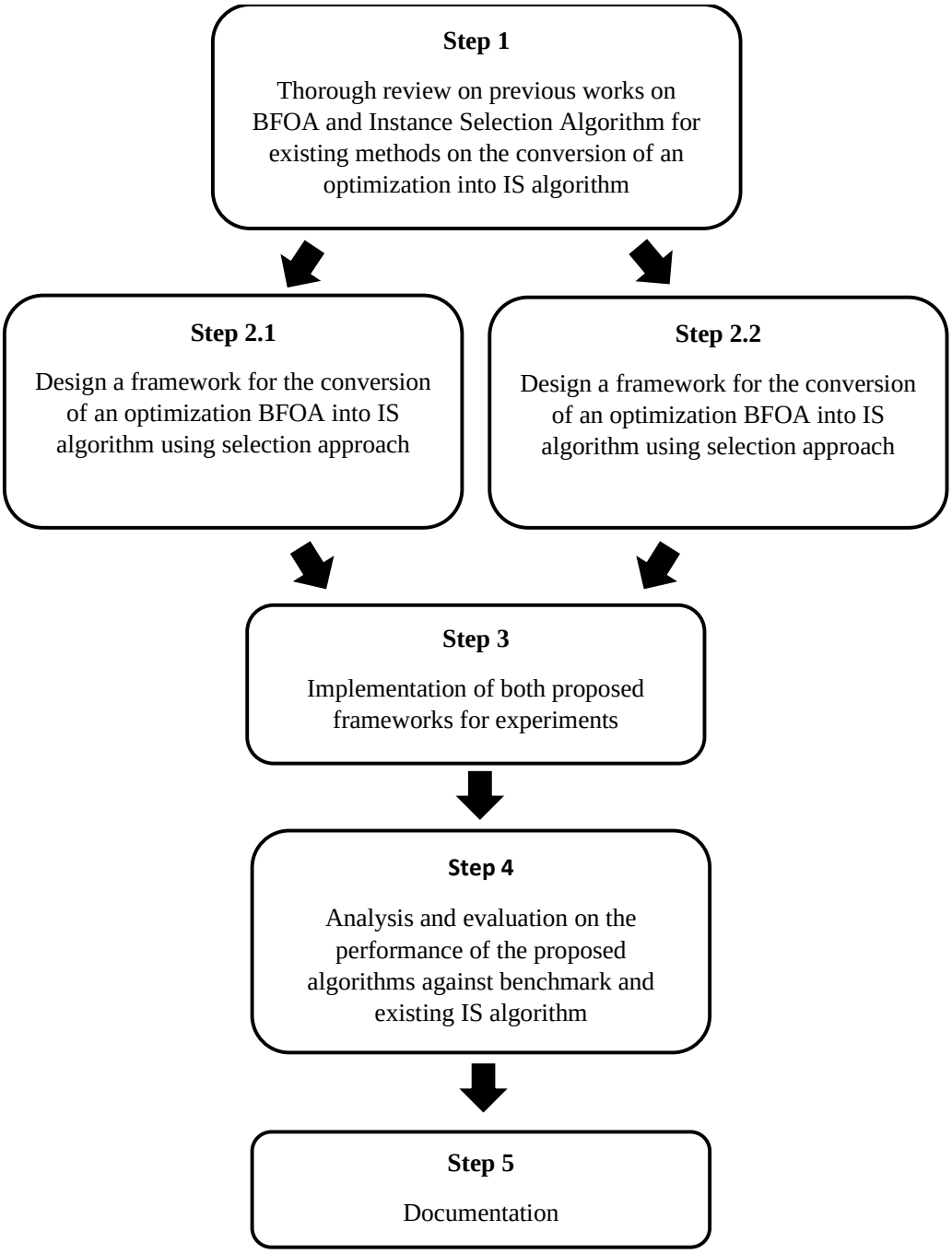


Figure 3.1: Research Methodology

Step 1 - Thorough reviews on previous study: This study focuses on the converting an optimisation algorithm BFOA into data classification algorithm specifically as Instance Selection algorithms. Therefore, this study begins with a review on original structures of Bacterial Foraging Optimisation Algorithm (BFOA). Then, the various existing Prototype

Selection (PS) and Prototype Generation (PG) algorithms especially the natured-inspired algorithm such as Genetic Algorithm (GA), Particle Swarm Optimisation (PSO) and Ant Colony Optimisation (ACO) are thoroughly reviewed to study on how those optimisation algorithms can be converted into IS algorithms. Those natured-inspired algorithms are meant for optimisation and they have successfully been adopted as instance selection (IS) classifiers either in PS or PG approaches. The important criteria for conversion of an optimization algorithm into an IS classifier are extracted and considered for designing the conversion frameworks of BFOA.

Step 2.1 - Design a framework for the conversion of an optimisation BFOA into IS algorithm using Prototype Selection approach: This phase concerned about designing and converting the original BFOA into an instance selection (IS) algorithm using prototype selection (PS) approach. Based on the reviews, there are some criteria need to be considered which are the solution representation and fitness function for the conversion of an optimization algorithm into IS algorithm. The design or structure of an algorithm is mostly affected by the structure of the data that they work on. In earliest work of evolutionary instance selection algorithm (Kuncheva & Bezdek, 1998), the authors proposed chromosomes like representation for the dataset to enable the Genetic Algorithm to process them. Without any modification of the original structure of GA, the dataset could not be processed, and classification task could not be produced. The authors clearly stated how the modification of GA make it possible to be employed as an IS algorithm with PS approach. In addition, some parameters are also suggested to control and improve the algorithm. Thus, this study used Kuncheva and Bezdek (1998) work as a main reference to develop an Instance Selection BFOA with PS approach since the study used a very basic and clear implementation with solution representation and fitness function for conversion of

optimization algorithm into IS algorithm with minimal modification. A framework will be designed for the conversion of BFOA into Prototype Selection algorithm is produced at the end of this phase that reflect to the first objective of the study.

Step 2.2 - Design a framework for the conversion of an optimisation BFOA into IS algorithm using prototype generation approach: This phase involved with designing the framework for the second version of BFOA for instance selection algorithm using prototype generation approach. As mentioned in the previous chapter, prototype generation is literally more dynamic than prototype selection because the data itself can be generated or fabricated. Due to this feature, the concept for the second proposed algorithm is quite different with the PS approach. According to literature, there are two ways for representation of the solution. The first representation is one population consist of many candidate solutions and the second representation is a population represented as a solution for a problem. The second representation method was chosen study to represent the solution for the proposed PG algorithm. This method is suitable with the dynamical nature of generation approach. The detailed framework and modification of the BFOA structure for the proposed PG algorithm will be discussed further in Chapter 4.3. The designed produced from this phase is reflected to the second objective of the study.

Step 3 - Implementation of both proposed frameworks for experiments: Based on the design that have been proposed, both proposed frameworks are implemented using MATLAB programming for experiment in this phase.

Step 4 - Analysis and Evaluation: This phase involved with the process of evaluating the performance of the proposed algorithms with existing Instance Selection (IS) algorithm in term of generalization ability, reduction rate and time efficiency. An

experimental study was conducted to observe and compare the performance of the proposed algorithms with some benchmark algorithms. A preparation must be made before executing the experiment. All datasets and its pre-processing method will be explained in this chapter.

Experimental Setup: There are three algorithms involved in the experiment which are BFOA-S, BFOA-G, the earliest evolutionary Instance Selection algorithm GA (IS-GA) (Kuncheva & Bezdek, 1998) and 1NN classifier. Each algorithm needs an optimised parameter configuration to perform well in classification.

Most of parameter configurations for IS-GA are adopted from its original work. Only parameter P_{ini} is adopted from Hossin et al. (2017) for IS-GA because the algorithm worked with variety size of datasets. On the other hand, there is no parameter setting for 1NN classifier since there is no parameter in it. These algorithms will be the benchmark algorithms for the experiment. Apart from that, a heuristic setting was conducted for BFOA-S and BFOA-G since there is no recommended optimised parameter for these proposed algorithms. The heuristic process is done using MATLAB where the proposed algorithm is executed with several chosen parameter's combination. It started with the smallest value of parameters with a selected dataset which is using small dataset for faster processing and result. Each combination of parameter is running 10 times using a similar fold and dataset. The value of each parameter is increased with 5 to observe any improvement in term of accuracy and size of prototypes. The average of 10 times for each combination are recorded and compared to see which value is suitable for a specific parameter. The effect of changes in parameter value on the results are also recorded and plotted as shown in Figure 3.2 and Figure 3.3 for each parameter settings that can be documented as a guideline for further heuristic setting for the proposed BFOA algorithms. Figure 3.2 is generated from MATLAB

based on chemotactic iteration and Figure 3.3 is plotted based on reproduction iteration. There final configurations for both algorithms are presented in Chapter 5 (Subsection 5.1).

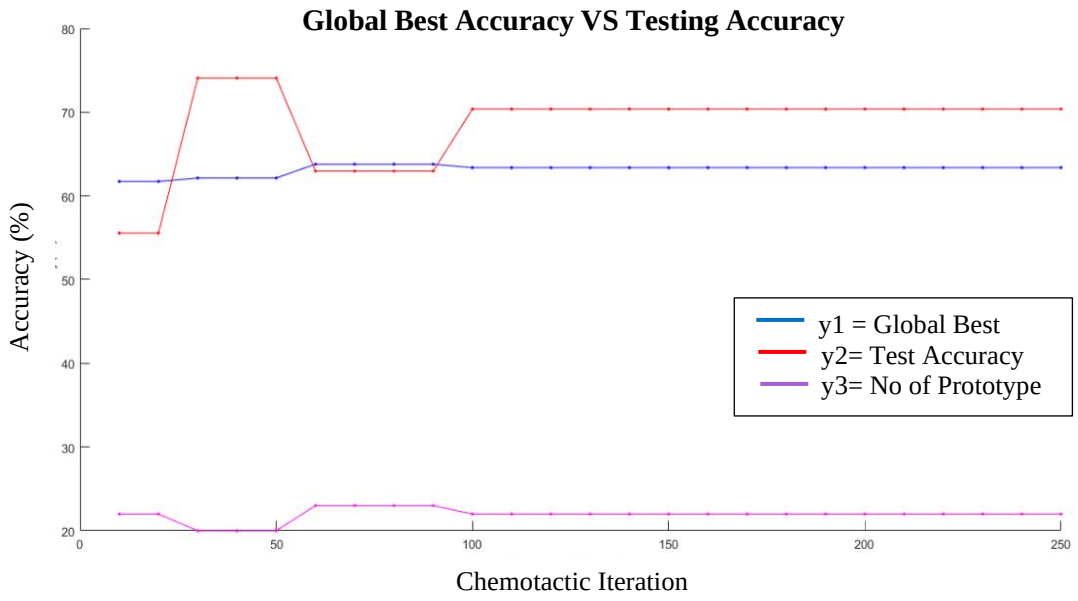


Figure 3.2: MATLAB Graph based on Chemotactic Iteration for Heuristic Setting.

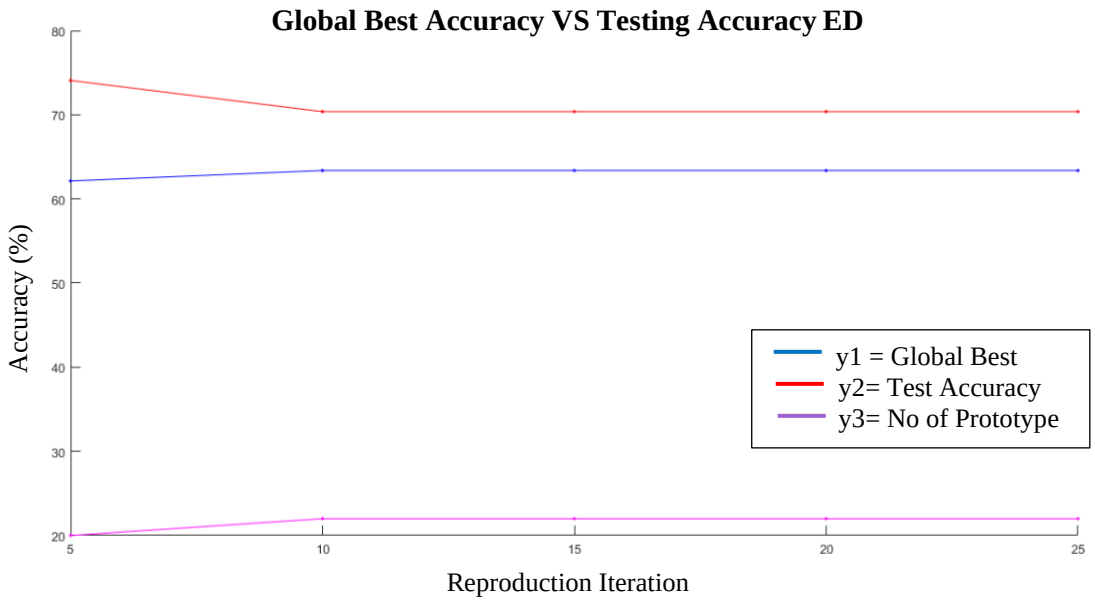


Figure 3.3: MATLAB Graph based on Reproduction Iteration for Heuristic Setting.

Evaluation: It is an inadequate to measure the generalization ability by taking the average values of the testing accuracy when it involved with comparing multiple classifiers using multiple datasets. It is hard to analyse them by relying only on the overall recognition values because in some cases, there is not so much different on the average values itself due to unnatural condition such as outliers in the testing results. To overcome the issue, two statistical tests were employed to evaluate and validate the results that are Sign Test and Wilcoxon Test. These two statistical tests were proposed by Demsar (2006) to evaluate the performance of multiple algorithms with multiple data. Furthermore, García-Pedrajas et al. (2010) and Garcia et al. (2012) also used similar evaluation methods to compare the performance between IS classifiers in their work. Through comparative descriptive statistics results obtained by these two studied measures, the proposed algorithms and the benchmark algorithm were compared using win-draw-loss number of datasets. The win, draw and loss record are counted by comparing each model in specific column with each model in each particular row. In this evaluation, win record is determined if a particular model in a particular column is performed better than the model in a particular row. Otherwise, the comparison result is counted as loss. Meanwhile draw (tie) is determined if both models in a particular column and row obtained the same result.

For further analysis, the sign test was applied to describe the significant difference between two models by interpreting the *p*-value based on two-tailed sign test. If the p-value of two-tailed sign test is below than 0.05 (5%), this study concluded that the two studied measures have significant difference. However, according to Demsar (2006), the sign test result has some limitations that makes the sign test results become less reliable. Therefore, the Wilcoxon test analysis is proposed as complementary test for comparing the two studied measures. In fact, this test is safer than the parametric test since the normal distributions or

homogeneity of variance can be ignored. Furthermore, this statistical test also can be applied to any specific criterion for comparison. Empirical results from (Demsar, 2006) have proven that non-parametric test is better than the parametric test for evaluation of multiple datasets over multiple classifiers. For easy understanding, this study also uses scatter plots to summarize the results obtained from 42 datasets that are categorized into small and medium datasets. Better result in the scatter plots are shown as the points are closer to the origin (0,0). It is easier to see which algorithms performs better in small and medium datasets.

This study also employed stratified 10-fold cross validation in the experiment to obtain the average of the training and testing error to measure the model produced by the algorithms (Garcia et al., 2010, 2012). It is used to measure the performance of the algorithm and it is reliable because the method can reduce variance and also bias. Each dataset is divided into approximately equal subset where *k*-1 is used for training and the remaining fold is used as testing dataset. Since the value of *k* is 10 for the experiment, thus the training and testing datasets are repeated for 10 times for each dataset and each algorithm. Then, the average of 10 independent experiments are taken and calculated to indicate and measure the performance of the algorithm. The stratification process is important to ensure that each fold has instances from every class so that each of them can be a good representative for the whole dataset.

Step 6 - Documentation: All designs, experimental results and analysis are collected and recorded in form of thesis and paper articles.

3.3 Datasets and Preprocessing Process

Various benchmark datasets were selected for the comparative experiments for the purpose of evaluation of the proposed algorithms. These datasets were obtained from the

UCI Machine Learning database (Frank & Asuncion, 2011). It consists of 19 binary class datasets and 23 multi-class datasets that varies in domain applications. In addition, there are a variation in term of relative proportions between each class in every dataset. Brief descriptions on the selected dataset are summarised in Table 3.1. There are three categories of the class proportion for the datasets which are balanced (B), imbalanced (IB) and extreme imbalanced (EIB). The dataset that has a minority class with 5% or less proportion compared to other classes will be classified as extreme imbalanced (Han et al., 2009). However, this study did not go further on the proportion of the classes because the focus is to evaluate the generalization capability and performance of the proposed algorithm on the various type of datasets. For the size of dataset (number of instances), it divided into three categories which are small (S), medium (M) and large (L). This specification is based on Garcia (2010), Garcia et al. (2012) and Triguero et al. (2012). Dataset with 2000 or less instances will be categorized into small dataset. Meanwhile, medium size dataset is ranged from 2001 to 20000 and dataset that more than 20000 is categorized as large. However, this study did not cover for the large volume of datasets due to the time complexity issue. For the attribute size categorization, this study followed the specifications from Chuang, Tsai and Yang, (2011). According to the authors, dataset with 19 or less attributes are considered as small dataset and attributes size less than 50 is considered a medium size. Attributes size more than 50 is classified as large size. However, the dataset with attributes size of 1000 and above is not included for the experiment due to the time complexity. In brief, there are 35 small size datasets and seven medium size datasets in term of volume or the number of instances. In term of size of attributes, 30 datasets are small, nine are medium and three are large size of attributes datasets selected for the experiments.

The selected dataset cannot be employed directly for the experiment purpose. There are some missing values present in some datasets. These missing values are replaced with median value for numeric value and mode for symbolic value that is similar to Skalak (1994) and Al-Shalabi (2011). Then, all datasets were normalized using min-max normalization method within the range of [0,1]. This normalization process is important to provide some standardisation and to avoid any attribute values from dominating the analysis (Al-Shalabi, 2011).

Table 3.1: Brief Description on 42 Benchmarks Datasets

Dataset	NoI	Size	Class	NoA	MV	CD	MnC(%)	MjC(%)
CCAus	690	S	2	14	No	IB	44.49	55.51
CCGer	1000	S	2	24	No	IB	30.00	70.00
Haberman	306	S	2	3	No	IB	26.47	73.53
Heart	270	S	2	13	No	IB	44.44	55.56
Hepa	155	S	2	19	Yes	IB	20.65	79.35
Ionos	351	S	2	34	No	IB	35.90	64.10
Liver	345	S	2	6	No	IB	42.03	57.97
Musk	476	S	2	166	No	IB	43.49	56.51
Pksn	195	S	2	22	No	IB	24.62	75.38
Pima	768	S	2	8	No	IB	34.90	65.10
Sonar	208	S	2	60	No	IB	46.63	53.37
SPECT	267	S	2	22	No	IB	20.60	79.40
SPECTF	267	S	2	44	No	IB	20.60	79.40
Threeof9	512	S	2	9	No	IB	46.48	53.52
Transfusion	748	S	2	4	No	IB	23.80	76.20
Votes	435	S	2	16	Yes	IB	38.62	61.38
WDBC	569	S	2	30	No	IB	37.26	62.74
WOBC	699	S	2	9	Yes	IB	34.48	65.52
WPBC	198	S	2	33	Yes	IB	23.74	76.26
Balance	625	S	3	4	No	IB	7.84	46.08
BreastT	106	S	6	9	No	IB	13.21	20.75
Cars	392	S	3	12	No	IB	17.35	62.50
Derma	366	S	6	33	Yes	IB	5.46	30.60
Glass	214	S	6	9	No	EIB	4.21	32.71
Hayes	160	S	3	4	No	IB	19.38	40.63
HeartC	303	S	5	13	Yes	EIB	4.29	54.13
Image	2310	M	7	19	No	B	14.29	14.29
Iris	150	S	3	4	No	B	33.33	33.33
LED7	3200	M	10	7	No	IB	8.44	10.66
Lenses	24	S	3	4	No	IB	16.67	62.50

Table 3.1 continued

Nthyroid	215	S	3	5	Yes	IB	13.95	69.77
OpticalD	5620	M	10	64	No	IB	9.86	10.16
PageB	5473	M	5	10	No	EIB	0.51	89.77
VerteC	310	S	3	6	No	IB	19.35	48.39
Vehicle	846	S	4	18	No	IB	23.52	25.77
Wave	5000	M	3	40	No	IB	33.06	33.84
Wine	178	S	3	13	No	IB	26.97	39.89
WQRed	1599	M	6	11	No	EIB	0.63	42.59
WQWhite	4898	M	7	11	No	EIB	0.10	44.86
Yeast	1484	S	10	8	No	EIB	0.34	30.86
Zoo	101	S	7	16	No	EIB	3.96	40.10

Note: NoI-no. of instances, NoA-no. of attributes, MV- missing value, CD-class distribution, MnC-minority class, MjC-majority class, S-small, M-medium, B-balanced, IB -imbalanced, EIB-extremely imbalanced, CCAus-Statlog Australian Credit approval, CCGer- Statlog German Credit, Haberman-Haberman's Survival data, Heart-Statlog Heart Disease, Hepa-Hepatitis Domain, Ionos-Johns Hopkins University Ionosphere data set, Liver-Bupa Liver Disorders, Musk-Musk Clean1 data, Pksn-Parkinsons disease, Pima- Pima Indian Diabetes, Sonar-Connectionist Bench (Sonar, Mines vs. Rocks), SPECT-SPECT heart data, SPECTF-SPECTF heart data, Threeof9-3 consecutive bits of 9 features are true, Transfusion-Blood Transfusion Service Center, Votes-1984 U.S Congressional Voting Records, WDBC-Wisconsin Diagnostic Breast Cancer, WOBC-Wisconsin Original Breast Cancer, WPBC-Wisconsin Prognostic Breast Cancer, Balance-Balance Scale, BreastT-Breast Tissue, Cars-Car Models data, Derma-DermatologyDatabase, Glass-Glass Identification, Hayes-Hayes Roth & & Hayes Roth (1977), HeartC-Heart Disease Cleveland, Image-Statlog Image Segmentation, Iris-Iris plants, LED7-LED display domain, Lenses-Database for fitting contact lenses, Nthyroid-Thyroid Gland Data, OpticalD-Optical Recognition of Handwritten Digits, PageB-Page Blocks Classification data set, Soya-Small Soyabean, VerteC-Vertebral Column, Vehicle-Statlog vehicle silhouettes, Wave-Waveform Databse Generator (Version 2), Wine-Wine Recognition data set, WQRed-Wine Quality (Red), WQWhite-Wine Quality (White), Yeast-Protein Localization Sites, Zoo-Zoo database.

3.4 System Requirements

The preprocessing process were done by using MS Excel 2016 as a tool. All processed datasets were presented as flats file and were saved under **.dat* extension. Besides, all the results and analysis were also presented using MS Excel 2016. The proposed algorithms were implemented using MATLAB 2016a and all experiments were running on a Dell Precision T3600 computer model. It is a workstation computer provided by Universiti Malaysia Sarawak. This computer is equipped with Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz and 16GB RAM. It was running with Windows 7 Professional Service Pack 1 operating system.

3.5 Summary

This chapter has presented the procedures and processes involved in conducting this study. It also provides information on how the results will be analysed and validated. In addition, the preparation of the benchmark datasets process is explained in this chapter to prevent any unreliable results and analysis at the end of the study. The necessary hardware and software are explained to conduct the experiment. In the next chapter, the implementation of the proposed frameworks will be explained in detail.

CHAPTER 4

PROPOSED ALGORITHMS

4.1 Overview

This chapter presents the conceptual designs of proposed BFOA for Prototype Selection and Prototype Generation algorithms together with their detailed implementation. All parameter configurations and experimental setup for both proposed algorithms were explained to evaluate the performance of the proposed algorithms.

4.2 Bacterial Foraging Optimisation Algorithm using Prototype Selection (BFOA-S)

The general implementation of BFOA for PS algorithm is portrayed in Figure 4.1. Through review, there are a lot of works on the conversion of nature-inspired optimisation algorithm into PS algorithm. However, there is no specific framework that converts an optimisation BFOA algorithm into a PS algorithm (classification algorithm). Thus, this study proposes a conceptual framework for the conversion of an optimisation BFOA into PS algorithm (accurately as evolutionary PS algorithm).

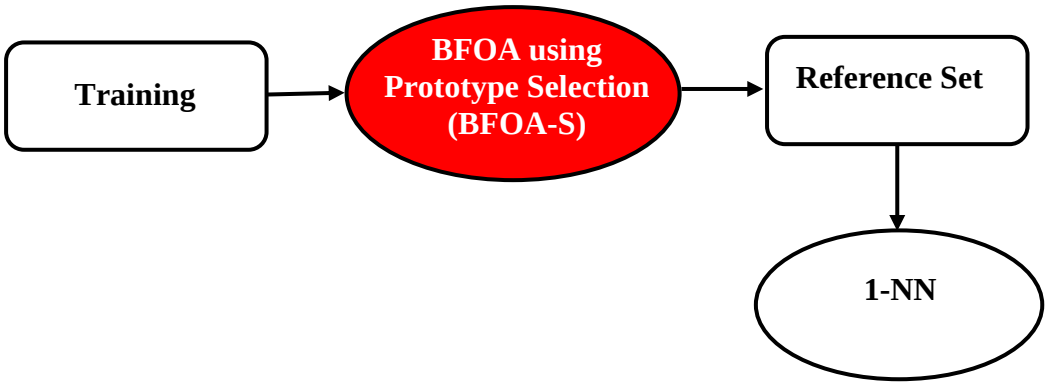


Figure 4.1: General Implementation of BFOA as a Prototype Selection Algorithm

The ultimate challenge in converting BFOA into PS algorithm is to design a good prototype set that can produce a reference set with minimal cardinality and simultaneously increase the classification performance of 1-NN. Another challenge is to retain the original mechanisms of BFOA (chemotaxis, tumbling, swimming, reproduction and elimination-dispersal) from the optimisation technique into the classification algorithm because improper modification will degrade its original capabilities. Apart from that, a proper solution and data representation are also important because they enable and affect the way of algorithm process the data.

From the literature, most of evolutionary PS algorithms are using method and guideline from Kuncheva and Bezdek (1998, 2001) to develop evolutionary PS algorithm. Since there is no framework exists for converting BFOA into PS, thus this study follows the guideline from Kuncheva and Bezdek (1998) to develop BFOA based PS algorithm. As stated in Chapter 2, it is compulsory to decide on representation of data and fitness function that going to be used to evaluate the candidate solutions first before modifying the targeted algorithm into PS algorithm. The details of the proposed framework to convert BFOA into PS algorithm is explained as follows.

4.2.1 Representation for BFOA-S

This study has employed Pittsburgh approach for solution representation. It is a common solution representation for PS algorithm. In Pittsburgh approach, an individual in a population will represent as a potential solution to a problem. Then, binary encoding method is used to encode information into the individual. This encoding method is widely used for the evolutionary PS algorithms.

In the proposed framework, an individual (a candidate reference set or potential solution) in the population is known as bacterium. Assume that each bacterium B_{all} be the set of given *j* instances $B_{all} = \{x_1, x_2, \dots, x_j\}$., where $j = 1,2, \dots, j$ are the data from *c* classes in a dataset *D*. Let *B* represents the selected data points and a reference set (solution), $B \in B_{all}$. Every bacterium in B is encoded with binary string that is based on the length j. Originally, each bacterium consists of two layers of string. The first layer is the index of instances in the dataset and the second layer is the binary string. The binary string consists the value of 0 to indicate inactive representative and 1 to mark as an active representative. The encoded binary string can be referred in Figure 4.2.

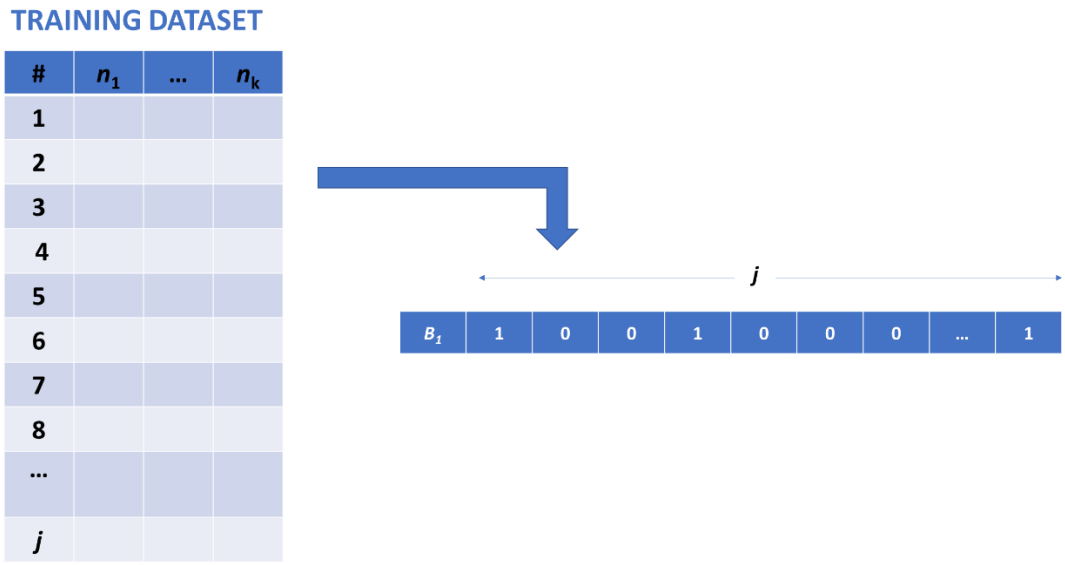


Figure 4.2: Data Representation using Binary Encoding Scheme

4.2.2 Fitness Function for BFOA-S

Classification accuracy is used as the fitness function to evaluate the candidate solutions in most classification problem. As the evolutionary PS algorithm also known as performance driven algorithm, it is natural to employ 1-NN classification accuracy to ensure that the algorithm become more sensitive to the presence of noise in the dataset. Classification accuracy can be interpreted as the ratio of correctly classified instances from x to overall number of instances j, $j=|x|$. However, the main objectives of PS are not fulfilled if the algorithm fails to find solution with the smallest cardinality even though the potential solution has better performance in classification. Both criteria are important in PS. Therefore, penalty function is adopted together with the classification accuracy to force the algorithm to select the best solution with smallest cardinality (Kuncheva & Bezdek, 1998).

The proposed algorithm needs to search for a set of *B** that satisfies $B^* = \arg\max_{Scx} F(B)$, where *F(B)* is the fitness function of the solution. The function *F* consists of two components as stated in Equation 4.1.

$$F(B) = A(B) - \propto f(|B|) \qquad \text{Equation 4.1}$$

The first component *A(B)* denotes the classification accuracy when using individual *B* as the potential solution. Then, the penalty function $\propto f(|B|)$ be the second component is applied to *A(B)* where $\propto f(|B|)$ denotes the cardinality function of *B* weighted by coefficient $\propto > 0$. The penalty value depends on the cardinality of the solution. The bigger size of solution, the higher the value of penalty that will be deducted from the accuracy. This action is to force the algorithm to prioritise the solution with smaller cardinality (Kuncheva & Bezdek, 1998). Parameter *T* is introduced to set the optimal cardinality threshold for a solution and the proposed algorithm needs to find those optimal solutions (Kuncheva & Bezdek, 1998). Penalty value is calculated based on the differences of the cardinality of solution compared to *T*.

4.2.3 Initialisation for BFOA-S

The initialisation process is started with the generation of the bacteria population as portrayed in Figure 4.3. A set of *N* bacteria in population *P* is randomly generated and represented as $P = B_1, \ldots, B_N$. Each bacterium has a binary string that can be referred as cells to represent the reference set. Each cell $B_N = \{x_1, x_2, \ldots, x_j\}$ value is generated and determined by the predefined probability S_{init}. This parameter is important to control either the string contain sparse or dense representatives (cell with the value of 1) (Kuncheva &

Bezdek, 1998). If the proposed algorithm intends to start with lower error rate, higher value of probability S_{init} is recommended. However, this study uses small value of S_{init}. It is purposely set to lower because the proposed framework intends to start with the lower cardinality of candidate solutions. This is consistent with the selected fitness function since the value of T is small, thus it is unpreferable to have higher penalty values from the beginning. Furthermore, it will make the convergence rate of the algorithm slower since the study aims for small cardinality and high classification performance reference set. Kuncheva and Bezdek (1998) recommends the value of probability of S_{init} is 10% or $S_{init} = 0.1$ from the dataset. However, this study employs a lower value S_{init}, that takes only 5% ($S_{init} = 0.05$) from the dataset. The reason is that the proposed algorithm will be working with small and medium size datasets. If 10% s instances are used as the representatives for the solution, then the cardinality of the solution become larger if the total size of the dataset is over than 2000 instances. Furthermore, the number of representatives is generated randomly, thus it may generate lower cardinality than 5% from the total size of dataset. Next section will explain on the modification of BFOA's main processes for PS algorithm.

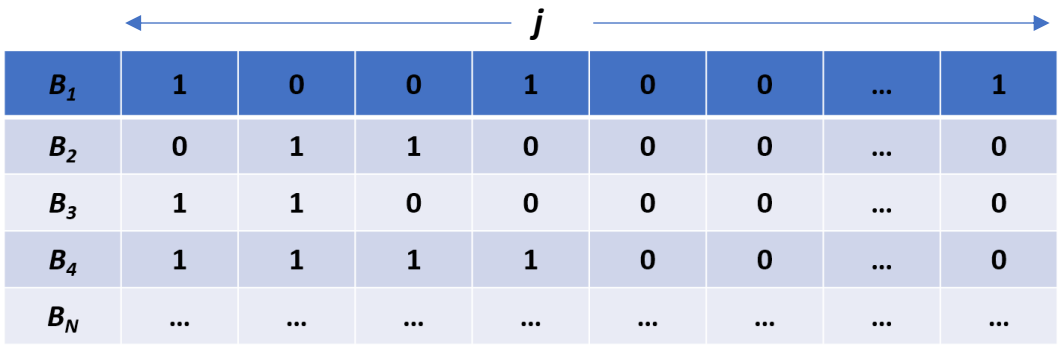


Figure 4.3: Population Initialisation of BFOA using Prototype Selection Approach

4.2.4 Chemotaxis for BFOA-S

Chemotaxis is the core process for local search operation. The proposed framework tries to preserve as much as possible the original concept of chemotaxis. Similar to the original concept of chemotaxis in BFOA, both subprocesses of swimming and tumbling are adopted in the proposed framework. However, the ways the work with the data are quite different from the original due to the differences in data representation. Each bacterium has two layers that represents the information of the dataset. The first layer consists of the index or position of an instances in the dataset and the other layer consists of binary values of 0 and 1. Chemotaxis process for the proposed framework will occur on the second layer only.

The way of chemotaxis process works in the proposed method is different compare the original optimisation chemotaxis. The original chemotaxis works by moving the bacteria around in the search space to find the optimal solution. However, the proposed method works with the cells within the bacteria which is different with the original concept. It tries to find the combination of representatives that can improve the classification accuracy. Representatives are the instances with the binary value of 1. Thus, chemotaxis will work with the representatives to find a good combination as the solution for the problem.

The proposed chemotaxis has two subprocesses called swimming and tumbling. Tumbling process will take place and followed by swimming process if the bacterium meets the minimum requirement of swimming process. These two modes are proposed to move representatives in the bacterium's cell. There are two directions for the movement of representatives. It is either moving forward ($+v$) or moving backward($-v$). The parameter v is introduced to control the length of the representative will move from its current position. Value of v is randomly generated from predefined range, v_{range}. The tumbling process starts

with the selection of a representative (instances that have value of 1 in the second layer) from the list of instances with binary value of 1. Next, a direction is randomly generated for the representative to move into a new position with random length of *v*. Figure 4.4 shows the process for tumble and swim process in the proposed algorithm. The details on swimming and tumbling processes are explained as follows.

Tumble: Assume that a representative has been selected randomly and it is expected to move into a new location with the direction to move forward (+) in a specific length of *v*. The current position of representative is mark with *i* and it will move into new position *i*+*v*. The algorithm will check the binary value of new position (*i*+*v*) either 1 or 0. If the binary value of the new position is 0 (inactive instance or not a representative), then it will be changed into 1 (a new representative). If it successfully changed into new representative, then the previous position will be loss its right as the representative. The value of binary of the previous position will change from 1 (active) into 0 (inactive). However, if the binary value of new position is 1, then algorithm will generate new length of *v* to search for new inactive instance. A successful tumble process will produce a new set of representative combination. This new combination is evaluated with the fitness function to measure its performance. The next process is triggered by this fitness value.

Swim: Swimming process is executed when the fitness value from the tumble process is higher than the previous fitness. Alternatively, this process can be described as a repetitive process of tumbling but in a similar length and direction. Swimming process will take place when the fitness function of new combination of representatives. $F(B_{s+1})$ after tumble is better than previous fitness function $F(B_s)$. The bacterium will keep swimming until a new

combination of representatives (reference set) $F(B_{s+1})$ produce inferior fitness value than the previous one or maximum swimming counter s is met.

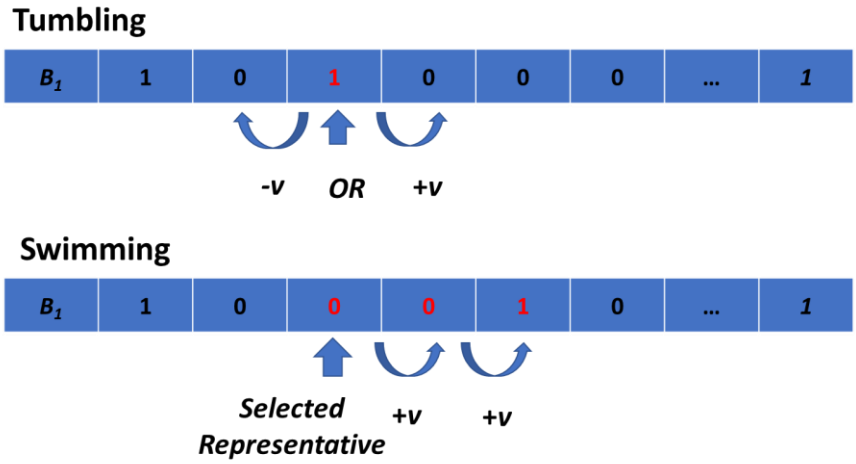


Figure 4.4: Tumble and Swim Process in Chemotaxis for BFOA-S

4.2.5 Reproduction for BFOA-S

Reproduction is responsible for the convergence of the algorithm for optimal solution. It is a process of replication and elimination of bacteria based on their health. The process is executed after chemotaxis process completes its cycles. The way of reproduction process works for the proposed framework is similar with the original reproduction process in optimisation BFOA. The population bacteria, P is sorted according to their fitness values. It starts from the healthiest (high value of fitness) bacterium and goes down into the poorest health (lowest value of fitness) of bacterium. Then, the bacteria will be divided in half into two groups. The first half consists of good health bacteria and the other half consists of poor health bacteria. The poor bacteria will be eliminated from the population and good bacteria will be replicated. Thus, a new population of good health bacteria is produced and will go through chemotaxis process. The process is repeated until maximum iteration N_{re} is met.

The whole reproduction process can be seen in Figure 4.5. This method will keep the number of bacteria constants throughout the whole process of the proposed algorithm.

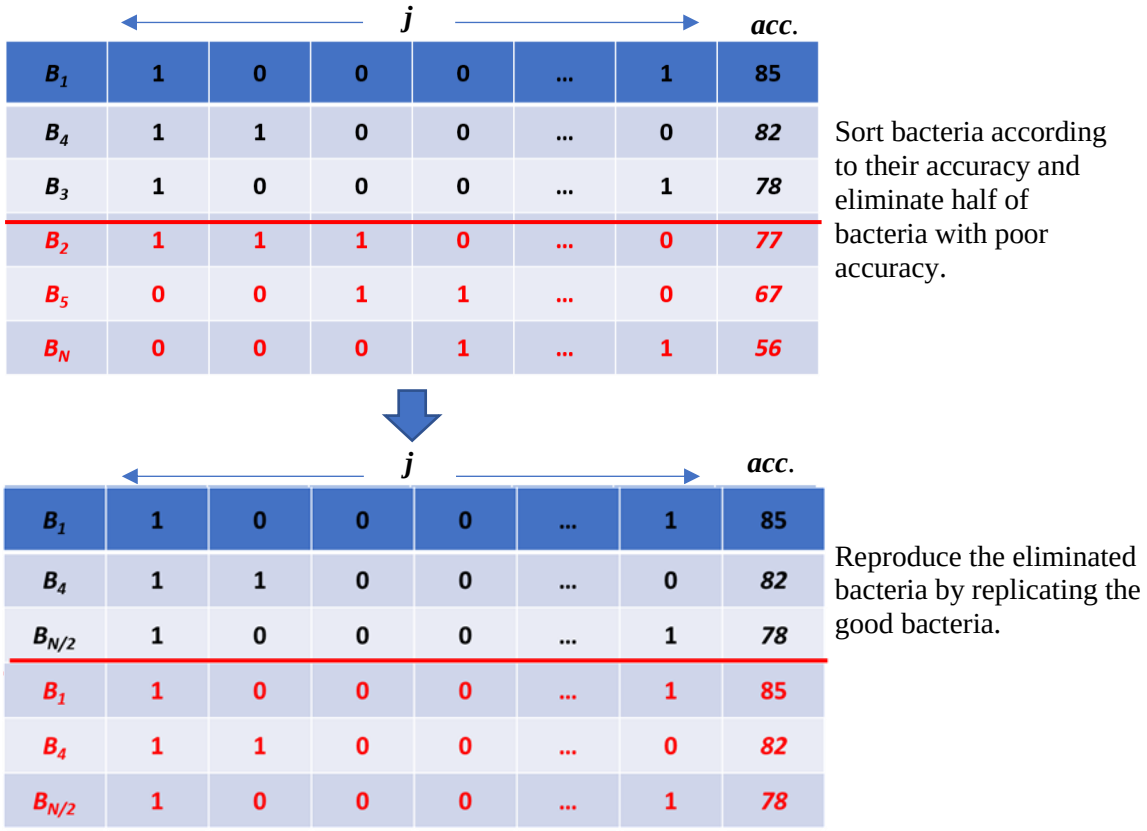


Figure 4.5: Reproduction Process for the Proposed BFOA-S

4.2.6 Elimination and Dispersal for BFOA-S

Elimination and dispersal is a process to help algorithm to expand its searching area and reposition the bacteria closer to the optimal solution. The process is also crucial to ensure that the algorithm does not trapped at local optima. This process contributes to the global search capability of BFOA. Technically, the process of elimination and dispersal involves with the process of reinitialization of bacteria with specific probability P_{ed} for the proposed framework. The control parameter P_{ed} is adopted as the external factor to trigger

the elimination and dispersal process. As portrays in Figure 4.6, once the elimination and dispersal event is triggered, all bacteria in the population are eliminated and reinitialized with new bacteria. These bacteria are the new individuals that consists of new combination of representatives. Then, they will be evaluated and proceed with chemotaxis and reproduction processes. Those processes will keep repeating until maximum iteration of elimination and dispersal, N_{ed} is met. There is situation where the elimination and dispersal is not triggered because of it does not meet the minimum requirement of P_{ed}. If this case happens, the population current population will remain unchanged and proceeds with chemotaxis and reproduction again until maximum iteration of N_{ed} is met. Complete pseudocode for BFOA-S can be referred in Figure 4.7.

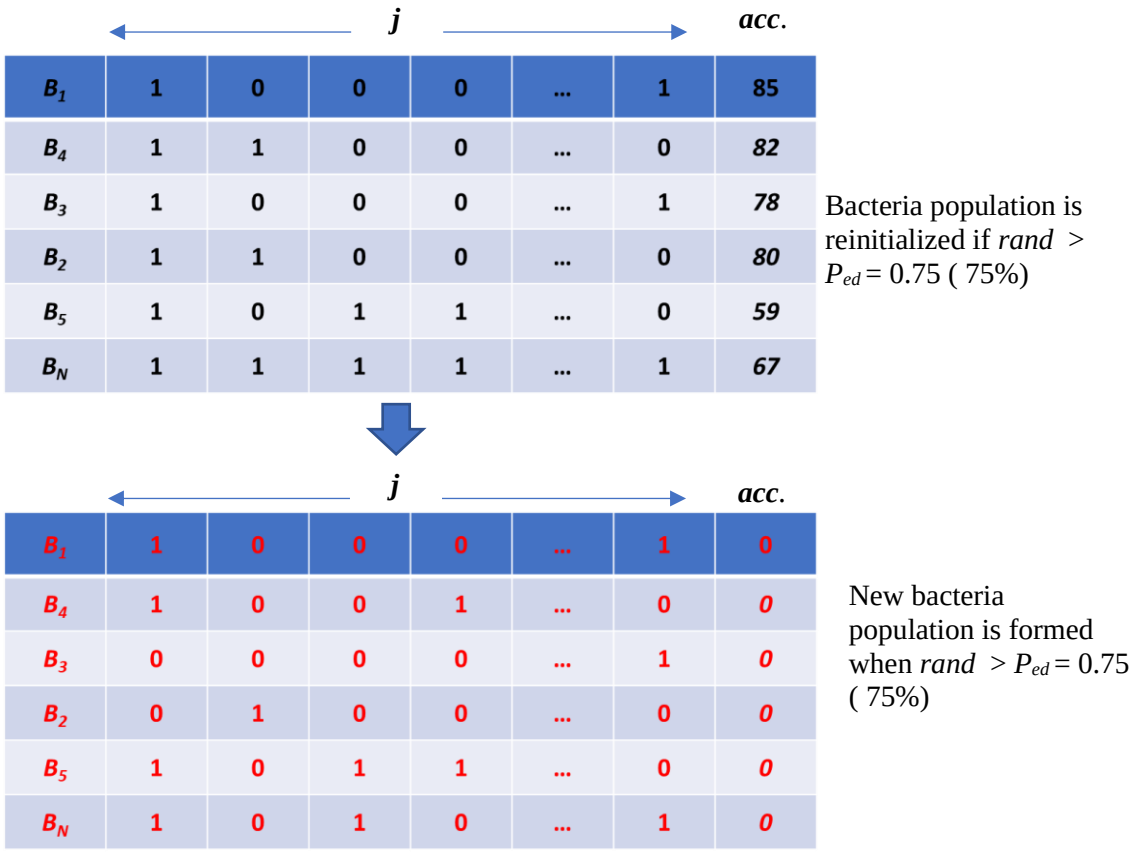


Figure 4.6: Elimination and Dispersal Process for the Proposed BFOA-S

[Step 1] Initialisation of parameters S, S_{init}, N_c, N_s, N_{re}, N_{ed}, P_{ed}, T, α, v_{range}

Bacteria loop: $i = i + 1$

a) Randomly select S_{init} from the training data as the representatives for bacterium i
b) Initialize localbestAcc, globalbestAcc

[Step 2] Elimination-dispersal loop: $l = l + 1$

[Step 3] Reproduction loop: $k = k + 1$

[Step 4] Chemotaxis loop: $j = j + 1$

a) for i = 1,2,…., S take the chemotactic steps for bacterium i
b) compute the fitness function for bacterium i
c) Save the values of *prevAcc = accuracy*, *prevPen = penalty*,
d) Compare the current accuracy with the local and global accuracy
   if *localbestPen* > *globalbestPen* then set
   globalbestAcc=localbestAcc;
   globalbestProt=localbestProt;
   globalbestPen=localbestPen;
   globalChro=localbestChro
   elseif localbestPen == globalbestPen, then check
   if *localbestProt* < *globalbestProt* , then set
   globalbestAcc=localbestAcc;
   globalbestProt=localbestProt;
   globalbestPen=localbestPen;
   globalChro=localbestChro.
e) Randomly select a representative in bacterium I and extract its position.
f) Tumble: generate a random direction (+ or -) and the length of chemostep, v within range [1, v_{range}]
g) Move the bacterium i from the current position into new position ($+v$ or $-v$).
h) Compute the fitness function of bacteria i with new combination of representatives.
i) Swim
   i. Let $m = 0$ (counter for swim length).
   ii. While $m < N_s$ (if have not swim too long).
   - Let $m=m+1$.
   - If *pen* > *prevPen* (better fitness), then repeat (g) using similar direction and length of v.
   - Else, let $m= N_s$. This is the end of the while statement.
j) Go to next bacterium (i+1) if $i \neq S$ (i.e., go to (b) to process the next bacterium).

[Step 5] If $j < N_c$, go to Step 4. In this case, continue the chemotaxis process since the life of the bacteria is not over.

[Step 6] Reproduction:

a) For the given k and l, and for each $i = 1,2,..., S$, let

$$J_{health}^{i} = \sum_{j+1}^{N_c+1} \text{Accuracy} \qquad (3)$$

Be the health of bacteria i (a measure of how many nutrients it got over its lifetime
and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic
the bacteria in order of ascending J_{health} (higher J_{health} means better health).

b) The S_r bacteria with the lower J_{health} values die and the remaining S_r bacteria with the best values will split (this process is performed by the copies that are made are placed at the same location as their parent).

c) The bacterium with highest J_{health} is compared with the previous global best solution or bacterium)

d) If current bacterium > global best bacterium, then replace the global best bacterium with the current bacterium

[Step 7] If $k < N_{re}$, go to step 3. In this case, the algorithm had not reached the number of specified reproduction step, so it starts the next generation of the chemotactic loop.

[Step 8] Elimination-dispersal: For $i = 1,2..., S$ with probability P_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse by reinitialize another one to a random search space. If $l < N_{ed}$, then go to step 2; otherwise end.

Figure 4.7: Pseudocode for BFOA-S

4.3 Bacterial Foraging Optimisation Algorithm using Prototype Generation (BFOA-G)

This section presents the conceptual framework for converting an optimisation BFOA into Prototype Generation (PG) algorithm. The challenges for the conversion of BFOA are similar with BFOA for PS algorithm since PG also a subcategory under Instance Selection. It is important to decide on the data representation and fitness function that is suitable with the concept of PG. Furthermore, PG concept is more flexible that it can modify and generate

new instances to produce better solution. Thus, the study needs to modify the BFOA appropriately to ensure that the modification will not reducing its original capabilities for BFOA as a PG algorithm. The general framework for prototype generation algorithm is shown in Figure 4.8 for the implementation of the proposed algorithm. The details for conversion of BFOA into a PG algorithm are described in the next section.

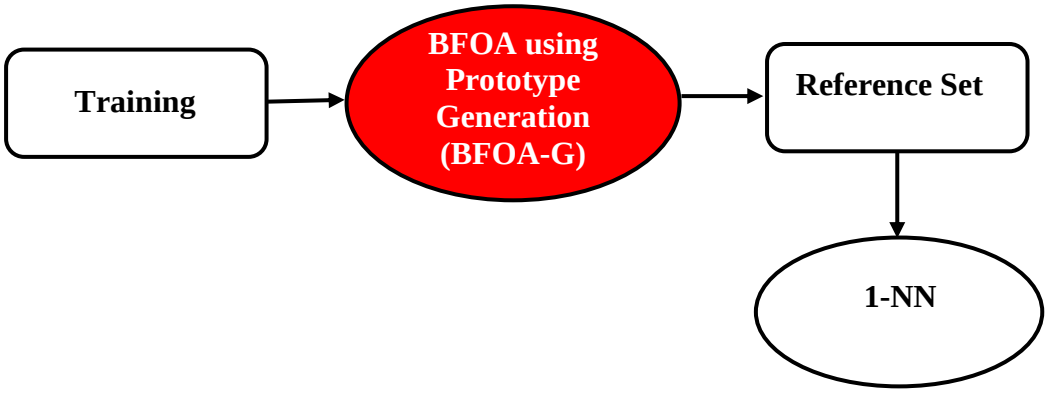


Figure 4.8: Prototype Generation Algorithm General Framework

4.3.1 Data Representation for BFOA-G

The proposed framework adopts data and solution representation from the Cervantes et al. (2009) to encode information into the solution. The authors introduce Michigan approach into PG algorithm for the nature-inspired Particle Swarm Optimisation (PSO) algorithm. In Michigan approach, only a part of solution is encoded into an individual. Therefore, a population of bacteria is representing a solution for a problem. The individual or prototype is called as bacterium in the proposed framework. As for the data representation value, real number is used since the attribute values from the datasets are used to represent a bacterium. Assume that $B \in B_{all}$ is a set instances with size of N that represent a solution. Given that B_{all} is a set of N number of instances where N is determined by the distribution of

instances in each class in the dataset. Given that bacteria $B_{all} = \{x_{(1,c)}, x_{(2,c)}, \ldots, x_{(N,c)}\}$ where x represents an instance with class label c. Each of x consists of the instance's attributes of n and a class of c. The class is assigned during the initialisation of the population and it remain unchanged even the algorithm completes its cycles. Figure 4.9 shows the representation format for a bacterium in a population.

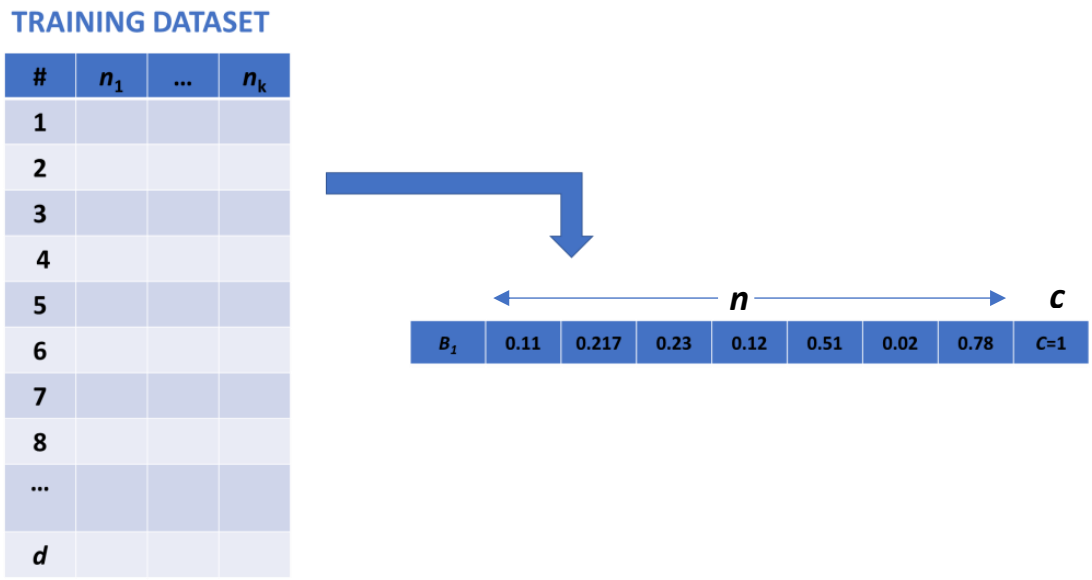


Figure 4.9: Data Representation for a Bacterium in BFOA-G

4.3.2 Fitness Function for BFOA-G

There are two fitness functions for the proposed framework. The proposed algorithm must generate a set of solution $*B_{all}$ that can maximize the classification performance of nearest neighbour classification. The objective can be denoted as $*B_{all} = \arg\max_{Scx} F(B_{all})$, where $F(B_{all})$ is the objective function. Since a bacterium only carries a part of solution in the proposed framework then a normal classification accuracy is unable to measure its performance. A fitness function called total correct prediction, *tcor* is employed to measure

the performance of a bacterium which is also an instance against all instances with the same class label. It can be denoted as shown in Equation 4.2.

$$tcor(x, c) = \frac{Total\ number\ of\ correct\ prediction\ of\ x\ with\ class\ c}{Total\ number\ of\ instances\ of\ class\ c} \quad \text{Equation 4.2}$$

However, this metric unable measure a solution's performance in classification. It is only working as health indicator for reproduction process. Classification accuracy is employed to evaluate the performance of the solution. It is a performance metric that calculates the ratio of correct prediction using B_{all} against total number of instances.

4.3.3 Initialisation for BFOA-G

Population initialisation is an important step to generate prototypes or individuals. The number of bacteria in a population is determined by the distribution of instances in each class. Assume that a population $P = \{ B_1, B_2, \ldots, B_N\}$ where *N* is the total number of bacteria in a population *P*. Let $B_N = \{x_{(1,c)},\ x_{(2,c)}, \ldots, x_{(N,c)}\}$ where *x* is an instance with a class label of c. Therefore, bacterium is consisted by a set of attributes with a class as shown in Figure 4.10. However, the total number of bacteria in a population, *N* is determined by a set of rules to maintain the distribution of dataset. The value of *N* will represent the cardinality of the solution. The cardinality will stay the same from the beginning until the end of the algorithm cycles. The rules are proposed as a scheme to reduce the cardinality of solution without ignoring the distribution of dataset. The rules for the determination of the total number of instances for each class is shown in Equation 4.3.

$$f(g_c) = \begin{cases} 1, & 0 \leq g_c \leq 4 \\ 2, & 5 \leq g_c \leq 10 \\ 3, & 11 \leq g_c \leq 20 \\ 5, & g_c > 20 \end{cases} \quad \text{where } g_c = \text{Total number of instances in class } c \text{ (no)}$$

Equation 4.3

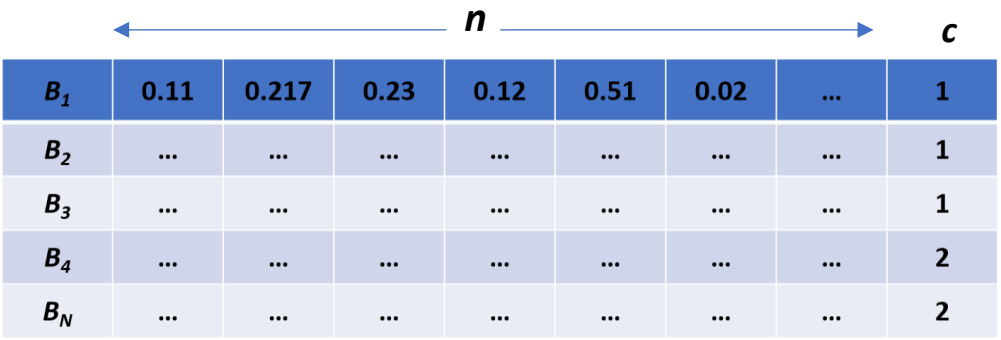


Figure 4.10: Bacteria Population Initialisation

4.3.4 Chemotaxis for BFOA-G

The proposed framework adopts chemotaxis with some modification. It is responsible for the local search capability of the proposed algorithm. All subprocesses in the original chemotaxis are preserved in term of concept for the proposed framework. Chemotaxis in the proposed framework has two subprocess named swim and tumble. The way of these processes work is depended on the representation of a bacterium. Since a bacterium consists of a set of attributes with a certain class, chemotaxis is executed by manipulating the value of the attributes in the bacterium's cells. There are two modes for the modification the selected attribute. It is either increasing (+) or decreasing (-) the attribute's value with specific chemostep *v*. The variable *v* is a random integer that is generated from a predefined range. Chemotaxis will start with tumbling process and follow by swimming process if the requirement for swimming is fulfilled. The details for both processes are explained as follows.

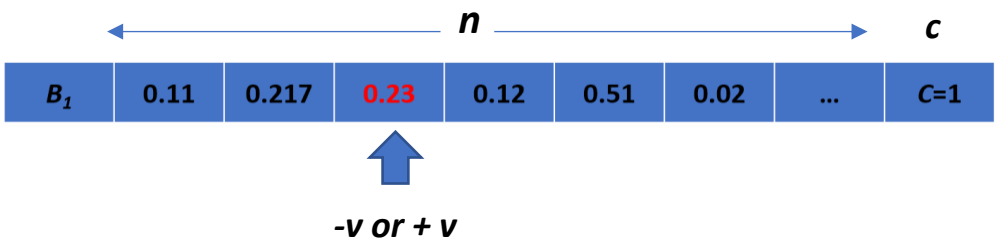


Figure 4.11: Tumble Process for a Bacterium in BFOA-G

Tumble: Tumble process starts with the selection of an attribute in bacterium B_N. The normalized value of the attribute is extracted for modification. Then, a direction and the value of chemostep *v* are generated randomly as shown in Figure 4.11. The predefined chemo step range, v_{range} is introduced to limit the generated value of *v*. For example, the algorithm decides on the increasing value of the attribute. The extracted value will be increased with the value of *v* (+*v*). Then, a mutated bacterium is produced. The fitness functions are applied to measure the classification performance of the population of bacteria as a single solution and also the individual performance of the bacterium against all instances with similar class label. The results are recorded but only the classification accuracy of the population of bacteria is considered for evaluation in chemotaxis. All bacteria will go through similar tumble process and they will be evaluated every time the bacterium's attribute is mutated with new value.

Swim: Swim process is executed if the fitness value of current solution P_{new} after tumble is better than the previous fitness function of solution P_{old}. It is the requirement for swimming process to be executed. Similar to the original concept of swimming process in optimisation BFOA, the swimming itself is a repetitive process of tumble with similar attribute, direction (mode) and chemostep *v*. The evaluation process for the bacteria population is similar with

tumble process. Both fitness functions are employed but only classification accuracy of the bacteria population as a single solution is considered for next swimming process. The swimming process will repeat until maximum iteration, N_s is met, or the current fitness value is inferior than the previous. The process can be referred in Figure 4.12.

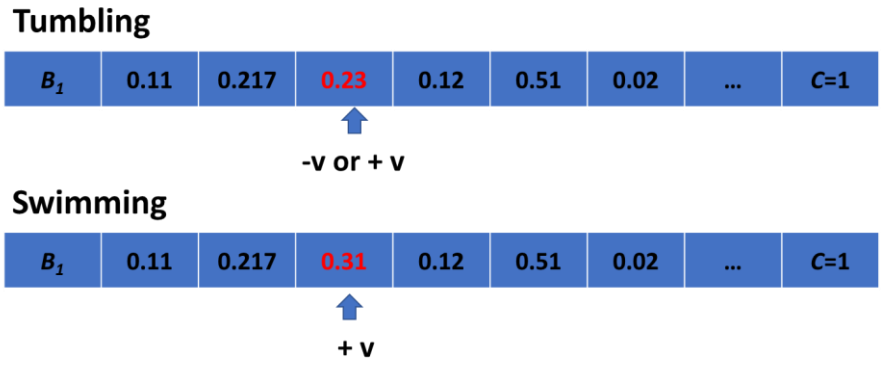


Figure 4.12: Tumble and Swim Process of a Bacterium for BFOA-G

4.3.5 Reproduction for BFOA-G

Reproduction is a process of elimination of weak bacteria and reproduce good bacteria to ensure only good bacteria will survive for the next cycle. This process is responsible for the convergence of the solution for the proposed algorithm. The proposed framework introduces a different way of implementation for reproduction process. It is due to the representation of the bacteria. A bacterium in the proposed framework only represents a part of the solution. The sorting process that based on the classification accuracy is unable to occur because the classification accuracy of solution does not represent a bacterium health or performance. Thus, the elimination process cannot be executed.

However, there is an alternative fitness function introduced to measure the health of a bacterium for BFOA-G. The percentage of the total correct predicted rate of bacterium

(basically a bacterium is also an instance) against all instances with similar class in the dataset, *tcor* is employed for the reproduction process. In the proposed framework, the elimination of the bacteria is based on the value of *tcor*. The *tcor* value is taken from the chemotaxis process where a bacterium is evaluated using accuracy against all dataset and also against data in its own class. A minimum threshold is introduced to determine either the bacterium is eliminated or survived for the next algorithm's cycle. When a bacterium with *tcor* value below than 50% (*tcor* < 0.5) is categorized as weak or poor health bacterium. This weak bacterium will be eliminated and reproduced by using any instance's set of attributes from similar class during reproduction process. The *tcor* value is set as 50% because if the value is too high, the total elimination of bacteria can be occurred which is similar to role of elimination and dispersal process. On top of that, the algorithm will be degraded into exhaustive search since it hard to converge since the population is totally changed for every iteration. This method will maintain the total number of bacteria in the population and ensure that there's no changes in the distribution of data in the solution.

It is important to maintain the distribution of dataset in the solution so that it can represent the dataset well. Improper reproduction process will produce an unacceptable solution. For example, assume that a bacteria population consists of bacterium with class A and class B. During the during reproduction process, it is found that all bacteria with class A have lower value of *tcor* and considered as poor health bacteria. Only bacteria with class B are considered as good bacteria. Improper elimination will erase the bacteria with class A completely and only bacteria with class B are remained in the population. Now the solution becomes smaller which is good for reduction rates, but it is only biased to class B. For the next iteration, the searching process is only resolved around class B and the algorithm will produce a final solution only for class B. This reference set is not acceptable to become the

solution because it will lead to misclassification if instances with class A are supplied. Thus, it is necessary to retain the ratio of instances for each class in the solution so that it will become an optimal solution for the classification problem. The reproduction process for the proposed algorithm can be referred in Figure 4.13.

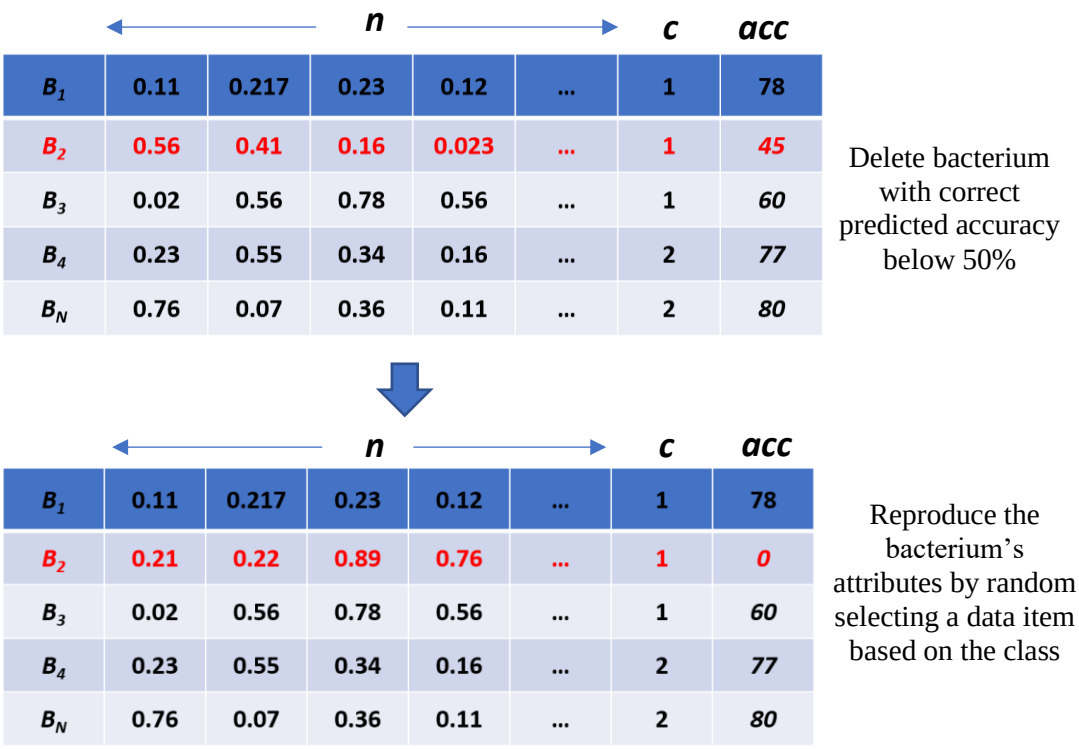


Figure 4.13: Reproduction Process for BFOA-G

4.3.6 Elimination and Dispersal for BFOA-G

Elimination and dispersal process are responsible for global search capability of the proposed algorithm. The implementation of elimination and dispersal process for the proposed framework is quite different compared to the original process in optimisation BFOA. However, the concept is preserved even though there is some modification on how the it works. This is due to the representation of the solution. In the optimisation BFOA, a

bacterium is eliminated and reinitialized into new place based on probability P_{ed}. For the proposed framework, elimination and dispersal occurs on a population of bacteria. The bacteria are eliminated and initialized again with the probability of P_{ed}. However, the bacteria are not eliminated completely but reinitialized the bacterium's attributes. The class assigned to each bacterium is unaffected to maintain the class distribution in the solution. Figure 4.14 portrays the elimination and dispersal process for the proposed algorithm. Once a population is reinitialized, it will undergo similar process such chemotaxis and reproduction process again. This process will keep repeating until maximum iteration of N_{ed} is met. Through this process, the algorithm able to explore in new search space and it also help the algorithm to avoid from trapped at the local solution. Full pseudocode for the proposed BFOA-G can be referred in Figure 4.15.

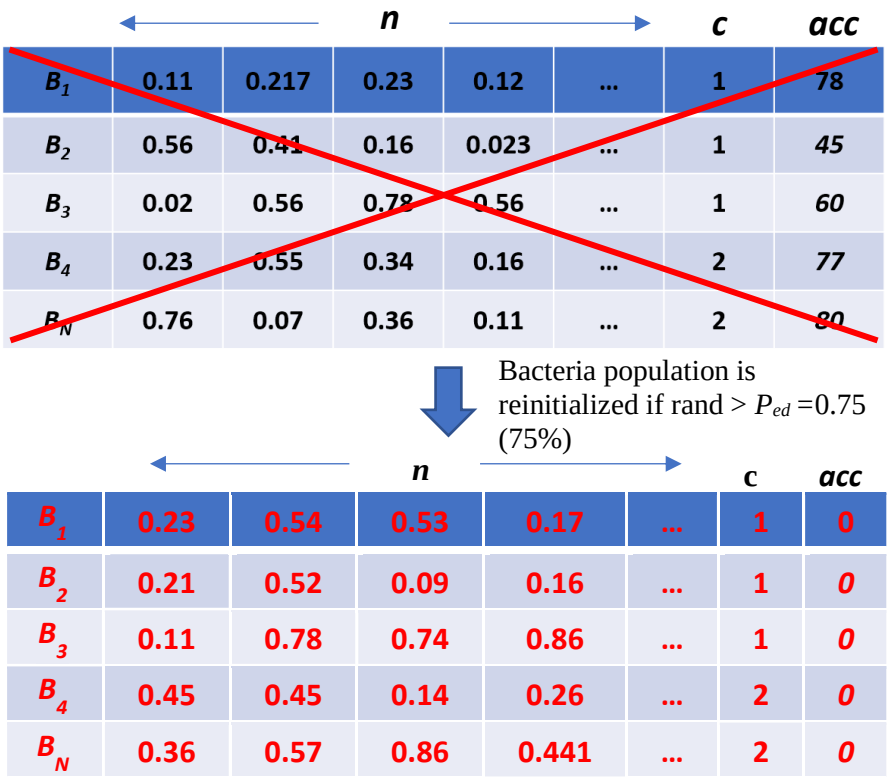


Figure 4.14: Elimination and Dispersal Process for BFOA-G

[Step 1] Initialisation of parameters S, S_{init} , N_c, N_s, N_{re}, N_{ed}, P_{ed}, v_{range}

Bacteria loop: $i = i + 1$

a) Randomly select S_{init} from the training data as the representatives for bacterium i
b) Initialize *localbestAcc, globalbestAcc*

[Step 2] Elimination-dispersal loop*:* $l = l + 1$
[Step 3] Reproduction loop: $k = k + 1$
[Step 4] Chemotaxis loop: $j = j + 1$

a) for $i = 1,2,\ldots, S$ take the chemotactic steps for bacterium i
b) compute the fitness function for bacterium i
c) Save the values of *prevAcc = accuracy, prevPen = penalty,*
d) Compare the current accuracy with the local and global accuracy
   if *localbestPen > globalbestPen* then set
   globalbestAcc=localbestAcc;
   globalbestProt=localbestProt;
   globalbestPen=localbestPen;
   globalChro=localbestChro
   elseif *localbestPen == globalbestPen*, then check
   if *localbestProt < globalbestProt* , then set
   globalbestAcc=localbestAcc;
   globalbestProt=localbestProt;
   globalbestPen=localbestPen;
   globalChro=localbestChro
e) Randomly select a representative in bacterium i and extract its position.
f) Tumble: generate a random direction (+ or -) and the length of chemostep, v within range [1, v_{range}]
g) Move the bacterium from the current position into new position ($+v$ or $-v$).
h) Compute the fitness function of bacteria i with new combination of representatives.
i) Swim
   i. Let $m = 0$ (counter for swim length).
   ii. While $m < N_s$ (if have not swim too long).
   - Let $m=m+1$.
   - If *pen* > prevPen (better fitness), then repeat (g) using similar direction and length of v.
   - Else, let $m= N_s$. This is the end of the while statement.
j) Go to next bacterium ($i+1$) if $i \neq S$ (i.e., go to (b) to process the next bacterium).

[Step 5] If $j < N_c$, go to Step 4. In this case, continue the chemotaxis process since the life of the bacteria is not over.

[Step 6] Reproduction:

a) For the given k and l, and for each i = 1,2,..., S , let

$$J_{health}^{i} = \sum_{j+1}^{N_c+1} \text{Accuracy} \qquad (3)$$

Be the health of bacteria i (a measure of how many nutrients it got over its lifetime
and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic
the bacteria in order of ascending J_{health} (higher J_{health} means better health).

b) The S_r bacteria with the lower J_{health} values die and the remaining S_r bacteria with the best values will split (this process is performed by the copies that are made are placed at the same location as their parent).

c) The bacterium with highest J_{health} is compared with the previous global best solution or bacterium)

d) If current bacterium > global best bacterium, then replace the global best bacterium with the current bacterium

[Step 7] If $k < N_{re}$, go to step 3. In this case, the algorithm had not reached the number of specified reproduction step, so it starts the next generation of the chemotactic loop.

[Step 8] Elimination-dispersal: if $rand > P_{ed}$

- Eliminate and disperse all bacteria in the population (this keeps the number of bacteria in the population constant).
- To do this, reset all attributes in all bacteria and repeat Step 1 to reinitialize the bacteria.
- If $l < N_{ed}$, then go to step 2; otherwise end.

Figure 4.15: Pseudocode for BFOA-G

4.4 Summary

As summary, this chapter presents the conceptual frameworks for the conversion an optimisation BFOA into PS and PG algorithm. Each of the framework for the proposed algorithms has their own method of data representation, chemotaxis, reproduction and elimination processes to enable the conversion from optimisation algorithm into PS and PG algorithm. Those frameworks were developed based on original concept of BFOA to avoid any degradation on its original capabilities to produce a good classification algorithm.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Overview

The objectives of this chapter are to present the findings of the heuristic settings for parameter configuration and the experimental study that has been conducted. The proposed algorithms BFOA-S and BFOA-G are compared and analysed with benchmark algorithms. The parameter configurations for both proposed algorithms are presented in this chapter. In addition, the proposed algorithms also compared against nine state-of-the-art Instance Selection algorithms in term of generalization ability and reduction rate. Relevant figures, graphs and tables are generated to explained and support the finding.

5.2 Parameter Configuration

An experimental study was conducted to observe and compare the performance of the proposed algorithms with some benchmark algorithms. A preparation must be made before executing the experiment. All datasets and its pre-processing method are already explained in Chapter 3. There are three algorithms involved in the experiment which are BFOA-S, BFOA-G, the earliest evolutionary Instance Selection algorithm GA (IS-GA) (Kuncheva & Bezdek, 1998) and 1NN classifier. Each algorithm needs an optimised parameter configuration to perform well in classification. There are several parameters that have been adopted and introduced in the proposed algorithms. All parameters for both proposed algorithms are listed in the Table 5.1.

Table 5.1: List of Parameters for the Proposed BFOA-S and BFOA-G

BFOA-S	BFOA-G
Bacteria size, S Chemotactic steps, N_c Swimming steps, N_s Reproduction steps, N_{re} Elimination and Dispersal steps, N_{ed} Probability of Elimination and Dispersal, P_{ed} Initial probability search, S_{init} Predefined cardinality of solution, T Weighting coefficient for penalty term, α Maximum range of Chemostep, v_{range}	Bacteria size, S Chemotactic steps, N_c Swimming steps, N_s Reproduction steps, N_{re} Elimination and Dispersal steps, N_{ed} Probability of Elimination and Dispersal, P_{ed} Maximum range of Chemostep, v_{range}

The parameter configuration from original BFOA cannot be simply adopted into the proposed algorithms. Thus, it is necessary to find the optimal parameter setting to ensure that the proposed algorithms can perform in optimal. A heuristic setting is employed for both algorithms to find the parameter configuration. The details of heuristic setting for BFOA-S and BFOA-G are explained as follows.

Heuristic setting for BFOA-S focusses on parameters for the core process of BFOA such as chemotaxis, reproduction and elimination and dispersal process. Some of parameters can be skipped for heuristic setting because there is some recommendation available for specific parameters by the previous researchers. The value of parameter S, $S_{init,}$ α, T are

adopted from Kuncheva and Bezdek (1998) and Hossin et al. (2017). The remaining parameter will undergo heuristic setting to find optimal values for each of them. Table 5.2 shows the list values for each parameter in heuristic setting. Those value are suggested due to the structure of the proposed BFOA algorithms. As shown in the previous section, the proposed algorithms consist of many nested loops. The creator of original BFOA (Passino, 2002) also suggests a lower value of parameter for the core processes in optimisation BFOA. The reason is to avoid the algorithm from consuming resources excessively to produce results. This study has produced a guideline and recommended a parameter configuration for BFOA-S from the heuristic result.

Table 5.2: Test Values for Each Parameter of BFOA-S in Heuristic Settings

Parameter	Test Values
N_c	5,10,15,20
N_s	5,10
N_{re}	5,10,15,20
N_{ed}	5,10
P_{ed}	0.25, 0.50, 0.75
v_{range}	5, 10

From the heuristic testing, the larger the value of chemotaxis steps, N_c will increase the time and complexity of the algorithm. However, if the value of N_c is too low, the algorithm may converge prematurely and can cause the algorithm trapped at local solution. The swim process does not affect directly to the performance of the algorithm since it has some requirement to be fulfilled before it can be executed. Bacterium may swim or not at all during a specific chemotaxis iteration and it depends on comparison on current fitness

against the previous fitness value. However, it helps the algorithm to search for a better solution. A suitable value of swim steps N_s is needed to avoid a bacterium swim for too long because it will cost more on the computational time of the algorithm. Apart from that, parameter v_{range} is also important to be initialized with an optimal value. It controls the distance of the bacterium will move during chemotaxis. v_{range} is a maximum range that the algorithm can generate for the movement of the bacteria. If the value is too low, the algorithm needs more iterations to obtain a good solution. However, if the maximum value is too large, the algorithm could miss the optimal solution during the run because the bacteria move too far than it should be.

Besides, when the value of chemotaxis steps N_c is optimal, the number of reproduction steps N_{re} will affect the convergence rate of the algorithm. The algorithm gains the capability to focus on good region and ignore bad region through reproduction process. The proposed algorithm may converge prematurely if the value of N_{re} is too low. However, if the value is too high, it will affect the computational time tremendously. Based on the heuristic testing, when the value of N_{re} is increased with the values in Table 5.2, the time complexity of the algorithm is doubled whenever the value is increased. Thus, the determination of the value of N_{re} must be made carefully to ensure the algorithm works efficiently.

The elimination and dispersal steps, N_{ed} is the outermost loop. Chemotaxis and reproduction reside in this loop. An iteration of elimination and dispersal will execute a complete iteration of Nc and N_{re}. As mentioned in the previous section, elimination and dispersal process is responsible for the global search capability of the algorithm. Lower value of N_{ed} is recommended because a single iteration of N_{ed} will affect tremendously on the time

complexity of the algorithm. Based on heuristic testing, lower value makes the algorithm produce result faster, but the result may not be closed to optimal. The algorithm may converge prematurely since it is not enough cycles to find a better solution. Furthermore, elimination and dispersal is controlled by probability P_{ed}. The elimination and dispersal process might occur or not at all in all iterations of N_{ed}. Higher value of P_{ed} indicates that more bacteria will remain in the population and small value of P_{ed} indicates that there is high chance for bacteria to be eliminated and dispersed into new search area. It is recommended to have slightly higher P_{ed} because the algorithm will be degraded into a random exhaustive search if there is too much changes in the bacteria population. However, the algorithm cannot have the maximum value of P_{ed} because the probability for the elimination and dispersal event to be occurred become zero. If there is no elimination and dispersal occur, then the algorithm will lose its global search capability to avoid from trapped at local solution. Thus, suitable value of P_{ed} is necessary to facilitate the algorithm to search for a better solution. From the heuristic testing, this study has come up with a parameter configuration for BFOA-S as presented in Table 5.3.

Table 5.3: Recommended Parameters Value for BFOA-S and BFOA-G from the Heuristic Settings

BFOA-S	BFOA-G
$S = 20$	$N_c = 10$
$N_c = 5$	$N_s = 10$
$N_s = 10$	$N_{re} = 20$
$N_{re} = 5$	$N_{ed} = 10$
$N_{ed} = 5$	$P_{ed} = 0.75$
$P_{ed} = 0.75$	$v = 0.1$
$T = 10$	
$\alpha = 0.1$	
$v = 5$	

Table 5.4: Test Values for Each Parameter of BFOA-G in Heuristic Settings

Parameter	Test Values
N_c	5,10,15,20
N_s	5,10
N_{re}	5,10,15,20,25,30
N_{ed}	5,10,15,20
P_{ed}	0.25, 0.50, 0.75
v_{range}	0.05, 0.10, 0.20

Table 5.4 shows the list of the values of BFOA-G's parameters for heuristic testing. The effect of the parameters is not so different compared to BFOA-S. Increasing in chemotaxis steps N_c value will increase the time and computational cost of the algorithm. Whenever the value of N_c increases within the listed values in Table 5.4, the time complexity

of the algorithm is doubled. In contrary, the algorithm unable to produce good result when the value of N_c is too low. It is a proof that the algorithm does not have enough iterations for the chemotaxis process to search for good solution. Therefore, it is necessary to find an optimal value for N_c so that the local search for solution can occur efficiently without wasting resources. Related to chemotaxis, the bacterium attributes are increased or decreased with random value generated within the range of parameter v_{range}. It is recommended to have lower value of v_{range} because the algorithm might miss the optimal value of the attribute if it is increased drastically. Furthermore, some attributes originally have higher value (value close to 1) so it is not possible to have high value of v_{range} because it will halt the chemotaxis process. However, the algorithm will need more iterations when the v_{range} is too low. It may lead to slow convergence rate of the algorithm.

The reproduction steps, N_{re} affect the convergence rate of the algorithm if all parameters in chemotaxis is optimal. It affects the capability of the algorithm to ignore the bad region and focus on the good region. The algorithm will converge prematurely if the value N_{re} is too small because the is not enough iteration to search for the best solution. In BFOA-G case, it preferable to have slightly moderate value of N_{re}. The computational cost of chemotaxis in BFOA-G is low due to small cardinality in every bacterium that is based on the number of attributes. On top of that, chemotaxis process works directly with the attribute values thus it does not cost much on the time complexity of the algorithm. However, the algorithm will not produce better result if the value of N_{re} is too high. Thus, it does not benefit the algorithm and only cost more on the resources.

Elimination and dispersal process is responsible for the global search capability of the algorithm. Based on heuristic testing, BFOA-G need more iterations to find a good

solution. A slightly moderate value for N_{ed} is recommended since chemotaxis and reproduction processes do not take too much time to complete their activities. Lower value makes the algorithm to produce result faster, but the result may not be closed to optimal. The algorithm may converge prematurely since it is not enough cycles to find a better solution. Furthermore, this process is controlled by probability P_{ed}. The elimination and dispersal process might occur or not at all in all iterations of N_{ed}. Higher value of P_{ed} indicates of lower chance for elimination and dispersal to occur. Contrary, small value of P_{ed} indicates that there is high chance for the bacteria population to be eliminated and dispersed into new search area. It is recommended to have slightly higher value of P_{ed} because the algorithm will be degraded into a random exhaustive search if there are too much changes in the bacteria population. However, the algorithm cannot have the maximum value of P_{ed} because the probability for the elimination and dispersal event to be occurred become zero. If there is no elimination and dispersal occur, then the algorithm will lose its global search capability to avoid from trapped at local solution. Thus, suitable value of P_{ed} is necessary to facilitate the algorithm to search for better solution. Table 5.3 shows the recommended parameter configuration for the proposed BFOA-G that based on the result of the heuristic testing.

The final parameter configurations for the all algorithms in the experiment are presented in Table 5.5. Most of parameter configurations for IS-GA are adopted from its original work. Only parameter P_{ini} is adopted from Hossin et al. (2017) for IS-GA because the algorithm worked with variety size of datasets in the experiment. The configuration used for IS-GA is shown in Table 5.5. On the other hand, there is no parameter setting for 1NN classifier since there is no parameter in it. IS-GA and 1NN became the benchmark algorithms for the experiment.

Table 5.5: Parameter Configuration for BFOA-S, BFOA-G and IS-GA for the experiment

BFOA-S	BFOA-G	IS-GA
S = 20	N_c = 10	S=20
N_c =5	N_s = 10	*Mrate*=0.025
N_s = 10	N_{re} = 20	α =0.1
N_{re} = 5	N_{ed} = 10	T=10
N_{ed} = 5	P_{ed} = 0.75	*Gen*=500
P_{ed} = 0.75	v = 0.1	
T = 10		
α = 0.1		
v = 5		

5.3 Experimental Result and Analysis

This study conducted an experiment to evaluate the performance of the proposed algorithms. As mentioned in the experimental setup, two algorithms were chosen as the benchmark to measure the capability of the proposed algorithms as the Instance Selection algorithm. The core evaluation metrics for Instance Selection algorithms are generalization capability, storage requirement (cardinality of the reference set) and time complexity of the algorithm. A good IS algorithm can produce a reference set that has high generalization ability (high classification accuracy), low storage requirement and time complexity of the algorithm.

The experimental results obtained are shown in the Table 5.6. The average training accuracy, testing accuracy, average storage requirement, reduction rate and average time taken for training process obtained from 10-cross validation method were recorded for further analysis. The average results for 42 datasets are presented in Table 5.6 that indicates as the rough indicator for the algorithm's performance. Only testing accuracy and storage

requirement were collected for 1-NN classifier since there is no training process involved. In additional, Rec*Acc has been used as an estimator of the capability for an IS classifier to consider the trade-off between reduction and classification accuracy (Garcia et al., 2012).

Table 5.6: Experimental Results for BFOA-S, BFOA-G, IS-GA and Time Complexcity and 1NN Algorithms based on Average Testing Error, Storage Requirement

Datasets	**BFOA-G**			**BFOA-S**			**IS-GA**			**1NN**
	Accuracy	Storage	Time	Accuracy	Storage	Time	Accuracy	Storage	Time	Accuracy
BCD	**0.9666**	**0.0176**	28.0	0.9244	0.0211	141.6	0.8858	0.0308	364.1	0.9227
BCO	**0.9786**	**0.0143**	16.2	0.9600	0.0169	85.0	0.9457	0.0309	219.8	0.9471
BCP	0.7718	**0.0505**	9.9	**0.7726**	**0.0505**	17.4	0.7574	0.0510	53.0	0.6061
CARDAUS	**0.8783**	**0.0145**	22.2	0.8435	0.0213	114.1	0.8058	0.0313	279.8	0.8174
CARDGER	**0.7420**	**0.0100**	43.8	0.6780	0.0171	310.1	0.6040	0.0318	991.1	0.6030
HABER	**0.7872**	0.0327	6.2	0.7451	**0.0324**	14.3	0.7091	0.0333	30.9	0.6437
HEART270	**0.8407**	**0.0370**	7.8	0.8296	0.0430	22.9	0.8148	0.0444	49.2	0.7741
HEPA	**0.8967**	0.0645	5.3	0.8850	0.0652	13.9	0.8721	**0.0529**	23.8	0.8383
IONOS	**0.8860**	**0.0285**	18.6	0.8775	0.0365	83.1	0.8061	0.0407	165.4	0.8804
LIVER	**0.7040**	**0.0290**	8.0	0.6377	0.0325	25.4	0.6003	0.0357	50.8	0.6003
MUSK11	0.7393	**0.0210**	91.1	0.7249	0.0248	419.0	0.6999	0.0326	1249.5	**0.7645**
PARKINSON	**0.8976**	**0.0513**	7.4	0.8968	0.0605	21.8	0.8711	0.0549	38.5	0.8921
PIMA	**0.7864**	**0.0130**	19.0	0.6967	0.0190	110.5	0.6354	0.0319	242.7	0.6757
SONAR	**0.7926**	**0.0481**	17.0	0.6929	0.0524	61.6	0.6014	0.0543	95.1	0.5769
SPECT	0.8462	**0.0375**	10.1	**0.8499**	0.0453	23.9	0.8425	0.0446	71.8	0.7487
SPECTF	0.8387	**0.0375**	16.9	**0.8802**	0.0378	46.6	0.8503	0.0419	121.2	0.8085
THREE9	**0.8009**	**0.0195**	15.0	0.7579	0.0287	57.2	0.7227	0.0344	128.0	0.5450
TRANS1	**0.7928**	**0.0134**	14.6	0.7701	0.0187	77.5	0.7085	0.0306	158.2	0.6001
VOTES	**0.9608**	**0.0230**	13.1	0.9172	0.0253	49.9	0.8619	0.0333	134.4	0.9126
BALANCE	**0.8047**	0.0240	14.6	0.7952	**0.0210**	62.9	0.7441	0.0322	119.7	0.7776
BREAST1	**0.7264**	0.1698	4.0	0.7073	0.1519	7.1	0.6609	**0.1019**	8.5	0.6009
CARS	**0.7576**	0.0383	11.1	0.7323	**0.0311**	31.1	0.7015	0.0349	68.5	0.6989
DERMA	**0.9618**	0.0765	43.3	0.6967	**0.0325**	76.4	0.6255	0.0415	174.8	0.6914
GLASS	0.6961	0.1075	9.8	**0.7476**	0.0607	16.5	0.6732	**0.0509**	26.6	0.7470
HAYES	0.7125	0.0938	4.0	0.7125	0.0725	7.9	0.6313	**0.0669**	11.7	**0.7188**
HEARTC	0.5615	0.0759	15.9	**0.5906**	**0.0353**	27.2	0.5674	0.0373	59.2	0.5117
IMAGE	0.8766	**0.0152**	215.7	0.7896	0.0208	1350.0	0.7939	0.0376	4136.2	**0.9182**
IRIS	**0.9733**	0.1000	3.6	**0.9733**	**0.0460**	6.9	0.9267	0.0513	11.0	0.8200
LED7	**0.7450**	**0.0156**	207.7	0.5516	0.0206	1486.0	0.5078	0.0381	3886.8	0.2694

Table 5.6 continued

Datasets	BFOA-G			BFOA-S			IS-GA			1NN
	Accuracy	Storage	Time	Accuracy	Storage	Time	Accuracy	Storage	Time	Accuracy
LENSES	**0.9167**	0.2292	0.6	**0.9167**	0.3833	0.8	0.6333	**0.1875**	1.1	0.8667
NTHYROID	**0.9818**	0.0698	5.5	0.9442	0.0540	12.4	0.8892	**0.0488**	20.4	0.9398
OPDIGIT	**0.9021**	**0.0089**	2196.2	0.4014	0.0232	24143.7	0.3915	0.0396	91599.8	0.4546
PAGE	0.9128	**0.0046**	263.2	0.9417	0.0223	4655.1	0.9393	0.0390	14070.3	**0.9567**
SBEANS	**1.0000**	0.1915	2.3	0.9750	0.1468	3.9	0.9750	**0.1106**	4.8	0.9750
VCOL31	**0.7839**	0.0484	8.4	0.7613	**0.0290**	22.2	0.6968	0.0394	40.9	0.6419
VEHICLE	0.6087	0.0236	48.2	0.4172	**0.0193**	224.1	0.3782	0.0323	560.6	**0.6182**
WAVE	**0.7444**	**0.0030**	456.9	0.3402	0.0222	17736.5	0.3176	0.0394	36787.0	0.3290
WINE	**0.9778**	0.0843	6.8	0.9444	0.0584	13.7	0.9382	**0.0517**	23.1	0.9317
WQRED1	0.4791	**0.0156**	82.1	0.5291	0.0184	508.8	0.4747	0.0339	1315.3	**0.6479**
WQWHITE1	0.4020	**0.0060**	292.6	0.4194	0.0232	4501.3	0.4300	0.0394	12061.2	**0.6519**
YEAST	**0.5061**	0.0300	93.0	0.4090	**0.0185**	376.7	0.3753	0.0351	922.5	0.4360
ZOO	**0.9700**	0.1733	4.4	0.9409	0.1366	9.7	0.8318	**0.1168**	10.6	0.9400
	0.8073	**0.0516**	**103.8**	**0.7518**	**0.0499**	**1356.6**	**0.7071**	**0.0487**	**4056.9**	**0.7214**

Besides, two statistical tests were carried out to analyse further on the results obtained on testing accuracy and storage requirement. The statistical results are presented in Table 5.7 and Table 5.8. The value of *Ss* indicates the comparative descriptive statistic on win/draw/loss records from pairwise comparison. The first value indicates the win record of the number of datasets that is the algorithm (classifier) in the corresponding column performs better than the algorithm (classifier) in the corresponding row. Draw indicates the numbers of datasets that the compared two algorithms have similar performances. The third value is the loss which is referring to the number of datasets that the algorithm in the corresponding column has inferior performance compared to the algorithm in the corresponding row.

Table 5.7: Average result for 42 datasets for BFOA-G, BFOA-S, IS-GA and 1-NN

Algorithm	**Average Testing Accuracy (Acc)**	**Average Storage Requirement**	**Average Time Taken(s)**	**Reduction Rate (Red)**	**Red* Acc**
BFOA-G	0.807339	0.051604	103.8	0.948396	0.765677
BFOA-S	0.7518376	0.049920	1356.6	0.950080	0.704711
IS-GA	0.707093	0.048749	4056.9	0.951251	0.672623
1-NN	0.721437	1.000000	-	-	-

Based on the experimental results presented in Table 5.7, it is clearly shown that both of the proposed BFOA-G and BFOA-S have better generalization capability (average testing accuracy) compared to the benchmark algorithms. The proposed BFOA-G significantly outperformed BFOA-S in term of generalization capability. This claim is supported by the statistical analysis in Table 5.8 where there is a significant difference between the proposed BFOA-G and BFOA-S. Besides, the statistical result in Table 5.8 also shows that there is a significant different between BFOA-G against GA and 1-NN in term of average testing

accuracy. Even though BFOA-S has inferior generalization capability compared to BFOA-G, it shown a better performance in term of classification accuracy against IS-GA and 1NN classifiers. It can be seen in Table 5.7 that the average testing accuracy of 42 datasets produced by BFOA-S is higher than IS-GA and 1NN classifiers. This finding is supported by the statistical analysis obtained in Table 5.8 that there is significant difference between BFOA-S against those two benchmark algorithms. In this case, it is clearly shown that both of proposed algorithms have higher generalization capability than the benchmark algorithms.

Table 5.8: Statistical Analysis for Testing Accuracy

Algorithm	**Analysis**	**Algorithm**			
		1-NN	**IS-GA**	**BFOA-S**	**BFOA-G**
1-NN	Ss Ps Pw	-	**17/1/24** 0.349 0.328	**31/1/10**** 0.002 0.009	**34/0/8**** 0.000 0.000
IS-GA	Ss Ps Pw	-	-	**39/1/2**** 0.000 0.000	**47/0/6**** 0.000 0.000
BFOA-S	Ss Ps Pw	-	-	-	**31/3/8**** 0.000 0.000
BFOA-G	Ss Ps Pw	-	-	-	-

Note: * the row algorithm has significant difference with the column algorithm for both statistical tests, **the column algorithm has significant difference with the row algorithm for both statistical tests

However, an IS algorithm is not considered an efficient algorithm if it needs high storage requirement to produce a high classification accuracy result. The core objective of the Instance Selection is to reduce the storage requirement while maintaining or increasing the classification accuracy. Hence, the average storage requirement is presented in Table 5.7 as a rough indicator for reduction rate capability of the algorithms. Table 5.7 shows that IS-

GA classifier has the smallest average storage requirement compared to BFOA-G and BFOA-S. It means that IS-GA has the highest reduction rate on the storage requirement compared to BFOA-G and BFOA-S. Surprisingly, the statistical analysis in Table 5.9 shows a contradicting result. The analysis result in the Table 5.9 shows that BFOA-S has better reduction capability (low storage requirement) compared to IS-GA in all 42 datasets. This can be seen in Table 5.9 that there is a significant difference between BFOA-S and IS-GA which is contradicting to the result presented in Table 5.7 for storage requirement. Besides, statistical analysis in Table 5.9 also shows that there is no significant difference between BFOA-G and IS-GA. Hence, the IS-GA does not outperform the proposed algorithms statistically. In fact, the proposed BFOA-S has the highest reduction capability (low storage requirement) on 42 datasets compared to IS-GA according to the statistical analysis. Apart from that, it can be seen in Table 5.9 that there is no significant different between the proposed BFOA-G and BFOA-S. Thus, they are not far different in term of reduction capability performance. Statistical analysis is not done for 1-NN because there is no reduction occurs in the algorithm.

Table 5.9: Statistical Analysis for Storage Requirement

Classifier	Analysis	Classifier			
		1-NN	**IS-GA**	**BFOA-S**	**BFOA-G**
1-NN	Ss Ps Pw	-	-	-	-
IS-GA	Ss Ps Pw	-	-	**28/0/14*** 0.003 0.020	**28/0/14** 0.045 0.731
BFOA-S	Ss Ps Pw	-	-	-	**25/1/16** 0.212 0.841
BFOA-G	Ss Ps Pw	-	-	-	-

Note: * the row algorithm has significant difference with the column algorithm for both statistical tests, **the column algorithm has significant difference with the row algorithm for both statistical tests

For deeper analysis on the capability of the proposed algorithms, the product of accuracy and reduction rate (Red*Acc) were calculated as presented in Table 5.7. It serves as an estimator for the capability of an IS classifier to balance the trade-off between reduction and classification accuracy. A higher value of Red*Acc indicates a better capability of balancing the trade-off between reduction rate and generalization capability (average testing accuracy). Based on the result, BFOA-G significantly outperforms BFOA-S and IS-GA in term of their capability to balance the trade-off between accuracy and reduction rate. Besides, the other proposed algorithm BFOA-S also showed a higher value of Red*Acc compared to BFOA-S. This means the BFOA-S has better capability to balance the trade-off between accuracy and reduction rate than IS-GA.

Another criterion to evaluate the algorithms is the time taken for the algorithm to complete its cycles to produce the final reference set. The average time taken is presented in Table 5.7 for all algorithms to indicate the time complexity of the algorithm. Based on the result, the proposed BFOA-G has the lowest average time taken compared to BFOA-S and

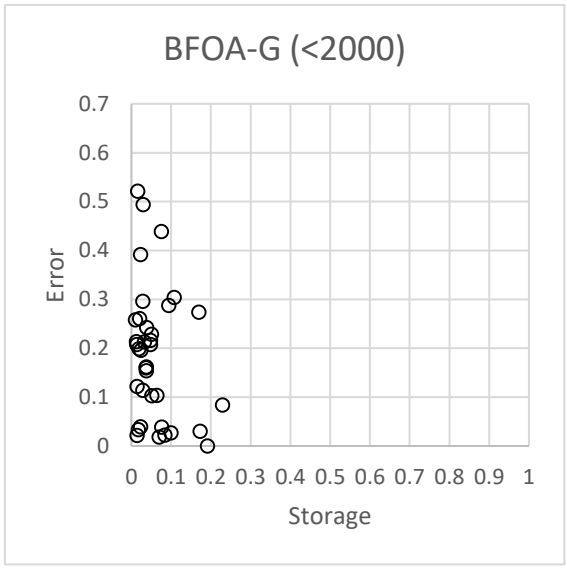
IS-GA. It shows that the proposed algorithm BFOA-G is the fastest and most efficient algorithm in the experiment. The difference of the average time taken for BFOA-G compared to BFOA-S and IS-GA is tremendous. BFOA-G is 13 times faster than BFOA-S and 39 times faster than IS-GA in completing its cycles. On the other hand, the proposed BFOA-S performed 3 times faster than IS-GA to complete its cycles. An interesting remark that can be pointed out is BFOA-S was adopted with similar initialisation method, fitness function and solution representation. However, there is huge difference in their time complexity. It can be claimed that the BFOA mechanism is more efficient compare to the GA mechanism in processing and producing the most relevant solution.

As the overall conclusion for the experimental study, both proposed algorithms BFOA-G and BFOA-S outperformed the performance of the benchmark IS-GA and 1NN algorithms in term of generalization capability, storage requirement, time complexity and the capability to balance the trade-off between reduction rate and generalization ability. The proposed BFOA-G is the most efficient algorithm in all evaluations for the experiment. Apart from that, BFOA-S also shows a good performance against the benchmark algorithm in all evaluation. The interesting finding is BFOA-S performed better in term of generalization capability, time complexity and on par with IS-GA in term of storage requirement. Both of algorithms share similar solution representation method, fitness function and initialization method but they produce result differently. Hence, it the capability of the original mechanism of the nature-inspired optimisation algorithm affects the performance of the proposed algorithm as an IS algorithm.

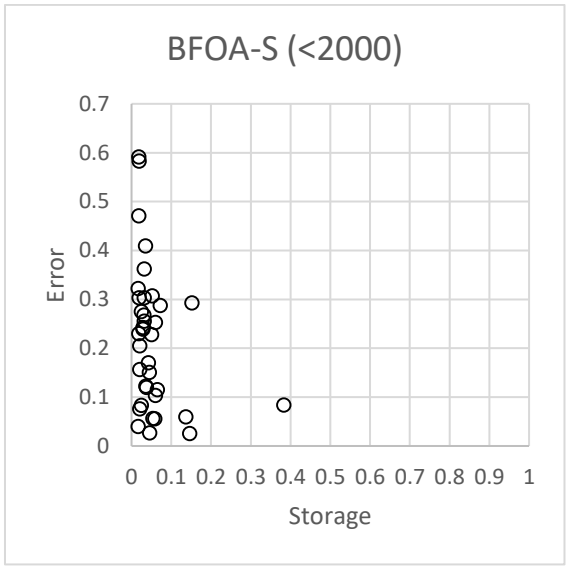
5.4 Graphical Summary for Small and Medium Datasets

A further observation has been made to study the behaviour of the proposed algorithms in small and medium datasets using the results attached in Table 5.6. A summary for small and medium datasets is plotted using scatter plot graph for each algorithm. It is to illustrate the relationship between the size of data and the generalization capability of the algorithm in tackling variety size of data. Small datasets are the dataset that contain less than 2000 instances and medium datasets contain more than 2000 but not more than 20000 instances. The details of the datasets can be referred in Chapter 3.

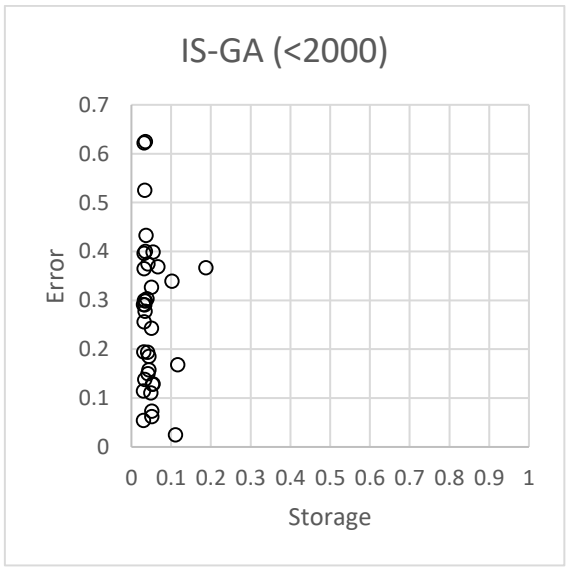
Figure 5.1 shows the generated scatter plots of small datasets for all algorithms and Figure 5.2 shows the generated scatter plots of medium datasets for all algorithms. These scatter plots are plotted based on the result of storage requirement (x-axis) against the testing error (y-axis) obtained from Table 5.6. Each of the scatter plot in Figure 5.1 contains 36 circles that represent 36 datasets with size of instances below than 2000. On the other hand, each of the scatter plot in Figure 5.2 contains 6 circles that represent 6 datasets with size of instances more than 2000 but less than 20000. A good scatter plot has more circles closer to origin (0,0). It can be translated as the reference set (model) has low testing error and storage requirement.



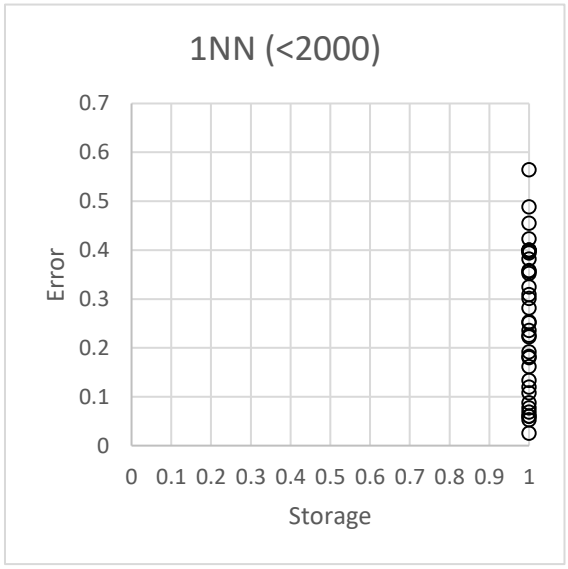
(a)



(b)



(c)



(d)

Figure 5.1: Scatter Plots of Storage Against Error for Small Datasets (a)BFOA-G(<2000), (b)BFOA-S(<2000), (c)IS-GA(<2000), (d)1NN(<2000)

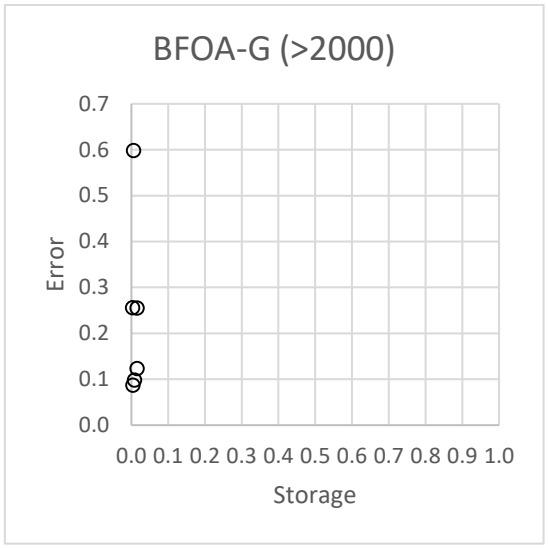
Based on scatter plots in Figure 5.1, several interesting patterns are obtained. The most outstanding pattern that can be seen is both of scatter plots by BFOA-G and BFOA-G has more circles concentrated closer to origin (≤ 0.3). This means that both proposed algorithms produced small cardinality of reference set (low storage requirement) with high

generalization capability (low error) for small datasets. Furthermore, both of scatter plots by BFOA-G and BFOA-S have circles that closer to origin on *x*-axis compared to IS-GA. It shows that both algorithms have better capability in searching for the smallest cardinality of reference set without reducing its generalization capability than IS-GA. It also can be seen in the comparison with 1NN that the proposed algorithms produce lower error reference set with small storage requirement compared to 1NN where 1NN used the whole dataset to produce result.

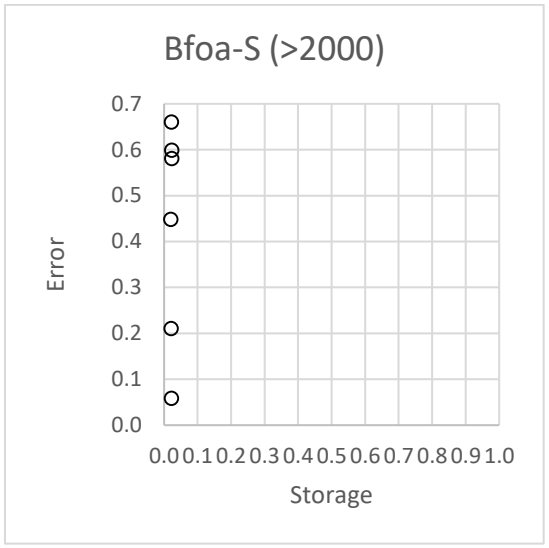
Another interesting observation that can be seen in Figure 5.1 is the scatter plots by BFOA-S and IS-GA. A closer look on scatter plot by BFOA-S shows an outlier spotted in the plot. This presence of outlier made the proposed BFOA-S look inferior to IS-GA since it can be misinterpreted as an incompetent IS algorithm in term of storage requirement. However, when the outlier in the scatter plot is extracted, it is from the smallest size dataset named as LENSES. This dataset consists of 24 instances with 3 classes. Based on the result in Table 5.6, BFOA-S obtained 0.3833 for storage requirement which is approximate to 9 instances and IS-GA obtained 0.1875 which is approximate to 5 instances. Then, the study extracts the error rate produced by both algorithms for LENSES dataset to see their classification performance. BFOA-S obtained 0.083333 (8.3% error rate) for the testing error and IS-GA obtained 0.366667 (36.7% error rate). This shows that BFOA-S prioritized the generalization capability and does not focussed on reduction. In contrary, IS-GA focusses more on reduction compared to the generalization capability of the reference set. In this case, BFOA-S performed better than IS-GA. For small dataset, differences with 4 instances will not give much impact on the time complexity of the algorithm.

Apart from that, an interesting pattern is observed in the scatter plots for medium datasets in Figure 5.2. Both of proposed BFOA-G and BFOA-S have circles closer to origin on *x*-axis compared to IS-GA. It shows that both algorithms have high reduction capability compared to IS-GA in larger size of datasets. Furthermore, the error rate produced by BFOA-G and BFOA-S were lower than IS-GA that can be seen in the scatter plots. Besides, BFOA-G has the lowest error rate and storage requirement in medium datasets. The real capability of BFOA-G can be studied when larger size of dataset is supplied. Out of 6 datasets, 5 of them are located below than 0.3 ($\leq$ 0.3) which is better than other algorithms in the experiments. Reduction capability is critical for medium datasets because it tremendously affect the time complexity of the algorithm when the size of reference is large. However, it must generate a high accuracy of reference set. High reduction without a good classification performance reference set is useless for IS algorithm because it contradicts with the IS concept. Consequently, poor classification performance is produced in latter process of classification.

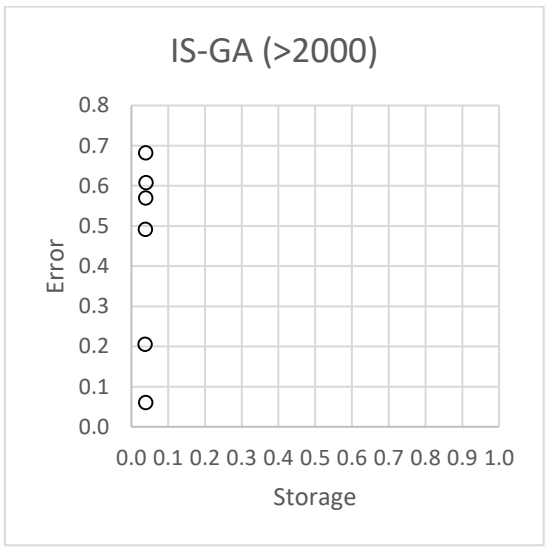
Another pattern that can be seen in Figure 5.2 is the pattern produced by scatter plots from BFOA-S and IS-GA. As mentioned above, BFOA-S and IS-GA share similar solution representation, fitness function and initialisation method. It relatable that the patterns produced by these algorithms are quite similar. They have a consistent reduction rate for medium dataset as can be seen on the *x*-axis in both scatter plots. However, it is clearly shown that BFOA-S has better performance compared to IS-GA since circles in scatter plot by BFOA-S are closer to the origin than circles in IS-GA's scatter plot. Compare to IS-GA, BFOA-S has lower classification error. Hence, BFOA-S outperformed IS-GA in both criteria.



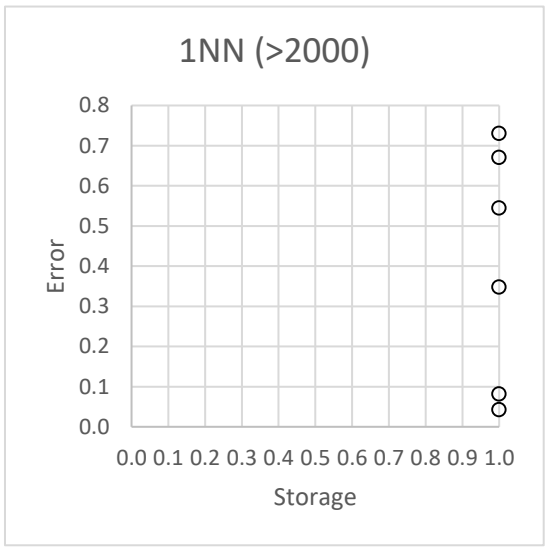
(a)



(b)



(c)



(d)

Figure 5.2: Scatter Plots of Storage Against Error for Medium Datasets (a)BFOA-G(>2000), (b)BFOA-S(>2000), (c)IS-GA(>2000), (d)1NN(>2000)

Apart from that, both proposed algorithms also produced competitive result against 1NN. Despite of using a partial of data to carry out classification task, both proposed algorithms able to produce good reference sets that are comparable when the whole dataset is used for classification. BFOA-G has better classification performance compared to 1NN

since more data are closer to origin compared to 1NN. This situation similar with BFOA-S that it also has lower error compare to 1NN for 6 medium datasets. Thus, both BFOA-S and BFOA-G able to outperform the performance of benchmark algorithms.

As overall summary for the graphical scatter plots, BFOA-G is the best IS algorithm for small and medium datasets. It has a better balancing capability between generalization accuracy and storage requirement compared to BFOA-S, IS-GA and KNN. Another proposed algorithm BFOA-S also has better balancing capability between generalization accuracy and storage requirement compared IS-GA and 1NN. Thus, both of algorithms outperformed the benchmark algorithms in small and medium datasets.

5.5 The Comparison Result with other IS Algorithms

The study has carried out an additional comparison and analysis between the proposed algorithms against the state-of-the art and recent IS algorithms. The results from Garcia-Pedrajas et al. (2010) were extracted for nine selected algorithms for the comparison. In addition, IS-GA is also included in this comparative study. 24 datasets were used for the comparison and analysis. The average storage requirement and testing error rate over 24 datasets were recorded and used for further analysis. The results can be referred in Table 5.10.

Table 5.10: The Average Results of Each Dataset for 12 Instance Selection Algorithms based on Testing Error and Storage Requirement

Data sets	IB3		DROP3		ICF		GA with CHC		MSS		EBIS	
	Stor.	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**
Balance	0.2426	0.2790	0.2179	**0.1694**	0.1862	0.2258	0.1277	0.2194	0.3169	0.2823	0.9235	0.2065
BCO	0.0310	0.0464	0.0638	0.0507	0.0375	0.0464	0.0549	**0.0421**	0.1013	0.0783	0.1771	0.0609
CardAus	0.1435	0.2594	0.2532	0.2319	0.2409	0.2087	0.0837	0.2667	0.3908	0.2652	0.2521	0.2580
CardGer	0.1461	0.3870	0.2559	0.2910	0.1880	0.3090	0.1447	0.3230	0.4309	0.3550	0.453	0.3110
Derma	0.1294	0.0889	0.1385	0.0667	0.1982	0.0694	0.0979	0.0861	0.1988	0.0917	0.2094	0.0917
Glass	0.3384	0.3381	0.2922	0.3238	0.2741	0.3333	0.1529	0.3762	0.4207	0.3143	0.6289	0.2714
Heart270	0.1214	0.2185	0.2111	0.2222	0.1745	0.2370	0.0802	0.2630	0.3716	0.2630	0.4946	0.2519
HeartC	0.1254	0.2467	0.2063	0.1800	0.2000	0.1867	0.1342	0.2567	0.3923	0.2600	0.3461	0.2734
Hepatitis	0.1029	0.2667	0.1479	0.2000	0.1379	0.2133	0.0814	0.2067	0.3186	0.2334	0.3267	0.2933
Ionosphere	0.1399	0.1086	0.0930	0.1457	0.0519	0.1514	0.1408	0.1371	0.2484	0.1457	0.1531	0.1229
Iris	0.1385	0.0867	0.1652	0.0600	0.3526	0.0800	0.0859	0.1067	0.2155	0.0933	0.5217	0.1000
Bupa-liver	0.1640	0.4265	0.4129	0.3971	0.3129	0.4265	0.0939	0.4471	0.5064	0.3971	0.9249	0.3706
New-thyroid	0.1077	0.0619	0.1273	0.0619	0.1057	0.0619	0.167	0.0571	0.1546	0.0476	0.5292	0.0524
Opdigits	0.0817	0.0651	0.1003	0.0514	0.0797	0.0792	0.4419	0.0342	0.1663	0.0425	0.502	**0.0215**
Pima	0.1280	0.3355	0.2225	0.2829	0.1942	0.2842	0.1292	0.3013	0.4142	0.3342	0.8507	0.3224
Sonar	0.1824	0.2150	0.3330	0.2750	0.2527	0.2750	0.1351	0.2300	0.3697	**0.1800**	0.1496	0.3550
Wine	0.1379	0.0647	0.1702	0.0588	0.1441	0.0470	0.1199	0.0647	0.1801	0.0765	0.3231	0.0823
Zoo	0.2110	0.0500	0.2319	0.1000	0.4736	0.0900	0.1626	0.0900	0.167	0.0700	0.1640	0.0900
Yeast	0.4142	0.5189	0.2718	0.4696	0.2474	0.4716	0.2017	0.4926	0.5339	**0.5230**	0.7232	0.4838
Image	0.1003	0.0697	0.1352	0.0641	0.1839	0.0745	0.0951	0.0762	0.1628	0.0498	0.4615	0.0615
Vehicle	0.3002	0.3321	0.3192	0.3036	0.2869	0.3107	0.1845	0.3440	0.4139	0.3321	0.6477	0.3012
Vote	0.0770	0.1047	0.0928	0.0907	0.1036	0.1070	0.0735	0.0884	0.1684	0.0930	0.1512	0.1372
Waveform	0.2851	0.3208	0.2709	0.2876	0.1838	0.3118	0.4956	0.2990	0.3435	0.3065	0.8755	0.2792
Page	0.0368	0.0761	0.0438	**0.0411**	0.0448	0.0415	0.226	0.0430	0.0991	0.0428	0.3578	0.0417
AVERAGE	**0.1619**	**0.2070**	**0.1990**	**0.1844**	**0.1940**	**0.1934**	**0.1546**	**0.2021**	**0.2952**	**0.2032**	**0.4644**	**0.2017**

Table 5.10 continued

Data sets	IMOEA		LVQPRU		CCIS		BFOA-S		BFOA-G		IS-GA	
	Stor.	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**	**Stor.**	**Te.Err**
Balance	0.2791	0.2952	0.1197	0.1903	0.0256	**0.1306**	**0.0210**	0.2048	0.0240	0.1953	0.0322	0.2559
BCO	0.1004	0.0580	0.0387	0.0377	**0.0138**	0.0333	0.0169	0.0400	0.0143	**0.0214**	0.0309	0.0543
CardAus	0.0432	0.2044	0.0644	0.2014	0.0201	0.1942	0.0213	0.1565	**0.0145**	**0.1217**	0.0313	0.1942
CardGer	0.0758	0.3240	0.0498	0.3030	0.0189	0.2700	0.0171	0.3220	**0.0100**	**0.2580**	0.0318	0.3960
Derma	0.0535	0.0528	0.5952	**0.0361**	0.0506	0.0583	**0.0325**	0.3033	0.0765	0.0382	0.0415	0.3745
Glass	0.0796	0.4476	0.5539	**0.3000**	0.0679	0.3143	0.0607	0.2524	0.1075	0.3039	**0.0509**	0.3268
Heart270	0.0701	0.3111	0.1951	0.2222	**0.0280**	0.2185	0.0430	0.1704	0.0370	**0.1593**	0.0444	0.1852
HeartC	0.0607	0.2967	0.1684	0.2567	**0.0243**	**0.1667**	0.0353	0.4094	0.0759	0.4385	0.0373	0.4326
Hepatitis	0.0281	0.2333	0.1607	0.2267	**0.0250**	0.2267	0.0652	0.1150	0.0645	**0.1033**	0.0529	0.1279
Ionosphere	0.0406	0.1457	0.1693	0.0971	0.0380	**0.0857**	0.0365	0.1225	**0.0285**	0.1140	0.0407	0.1939
Iris	**0.0266**	0.1667	0.3837	0.0733	0.0422	0.0733	0.0460	**0.0267**	0.1000	**0.0267**	0.0513	0.0733
Bupa-liver	0.0889	0.4147	0.1997	0.3971	0.0476	0.3647	0.0325	0.3623	**0.0290**	**0.2960**	0.0357	0.3997
New-thyroid	0.0857	0.0571	0.1763	0.0524	**0.0402**	0.0476	0.0540	0.0558	0.0698	**0.0182**	0.0488	0.1108
Opdigits	0.1450	0.0374	0.0234	0.0634	0.1602	0.0452	0.0232	0.5986	**0.0089**	0.0979	0.0396	0.6085
Pima	0.0544	0.3948	0.0708	0.2855	0.0157	0.2579	0.0190	0.3033	**0.0130**	**0.2136**	0.0319	0.3646
Sonar	**0.0460**	0.205	0.1591	0.2150	0.0739	0.2750	0.0524	0.3071	0.0481	0.2074	0.0543	0.3986
Wine	**0.0376**	0.1412	0.2429	0.0412	0.0379	0.0235	0.0584	0.0556	0.0843	**0.0222**	0.0517	0.0618
Zoo	**0.0349**	0.1000	0.4242	0.0500	0.0978	0.0800	0.1366	0.0591	0.1733	**0.0300**	0.1168	0.1682
Yeast	0.1790	0.5500	0.1424	0.4311	0.0276	**0.4007**	**0.0185**	0.5910	0.0300	0.4939	0.0351	0.6247
Image	0.0219	0.5282	0.0435	0.0541	0.1399	0.0853	0.0208	0.2104	**0.0152**	0.1234	0.0376	0.2061
Vehicle	0.0834	0.3310	0.0906	**0.2857**	0.0551	0.3298	**0.0193**	0.5828	0.0236	0.3913	0.0323	0.6218
Vote	0.0503	0.0489	0.0357	0.0930	**0.0161**	0.0605	0.0253	0.0828	0.0230	**0.0392**	0.0333	0.1381
Waveform	0.0856	0.2516	0.0120	**0.1796**	0.3634	0.3060	0.0222	0.6598	**0.0030**	0.2556	0.0394	0.6824
Page	0.1397	0.0441	0.0170	0.0431	0.0713	0.0457	0.0223	0.0583	**0.0046**	0.0872	0.0390	0.0607
AVERAGE	**0.0796**	**0.2350**	**0.1724**	**0.1723**	**0.0625**	**0.1706**	**0.0375**	**0.2521**	**0.0449**	**0.1690**	**0.0434**	**0.2942**

Table 5.11 presents the average result on testing errors and storage requirement over 24 selected datasets for all algorithms. Meanwhile, Table 5.12 and Table 5.13 show the statistical results of the proposed algorithms against ten IS algorithms in term of testing error and storage requirement. It is clearly shown in Table 5.11 that the proposed BFOA-G performed better against the pioneer IS-GA, three state-of-the-arts algorithms (IB3, DROP3 and ICF) and six recent algorithms by having the lowest average testing error. Apart from that, both of proposed algorithms obtained lower average storage requirement than the other 10 algorithms. BFOA-S dominates the ranking by having the lowest average storage requirement. Unfortunately, the result shows that BFOA-S has inferior classification performance compared to the recent algorithms such LVQPRU and CCIS in term of average testing error. BFOA-S only manages to outperform the benchmark IS-GA in term of average testing error over 24 datasets.

Table 5.11: Average Error Rate and Average Storage Requirement for 12 Algorithms

IS Algorithm	Average Error	Average Storage
IB3	0.206958	0.161892
DROP3	0.184383	0.199033
ICF	0.193413	0.193962
GA-CHC	0.202138	0.154596
MSS	0.203221	0.295238
Ebis	0.201658	0.464442
IMOEA	0.234979	0.079588
LVQ-PRU	**0.172321*****	0.172354
CCIS	**0.170562****	0.062546
IS-GA	0.294192	**0.043362****
BFOA-S	0.264096	**0.037500***
BFOA-G	**0.169008***	**0.044938*****

Note: *winner, **first runner-up, ***second runner-up

Statistical results in Table 5.12 show that the proposed BFOA-G significantly outperformed all eight algorithms except LVQPRU and $CCIS_{10}$. There are significant different obtained by BFOA-G for both statistical test against all eight algorithms in term of testing error. BFOA-G shows a competitive by having more win records against LVQPRU and $CCIS_{10.}$ However, the statistical result shows that there is no significant difference at confidence level 95% between BFOA-G against LVQPRU and $CCIS_{10.}$ Hence, they are competitive to each other. On the hand, the proposed BFOA-S also competitive against the other algorithms in term of testing error. The statistical result in Table 5.12 shows that there is no significant different between BFOA-S and other nine algorithm except IS-GA. On top of that, the statistical result also shows that the other algorithm unable to outperform BFOA-S statistically. As can be seen in Table 5.12, BFOA-S unable to outperform LVQPRU and $CCIS_{10}$ in term of win-draw-loss record but the statistical result also shows that LVQPRU and $CCIS_{10}$ do not have significant difference at 95% confidence level for both statistical tests against BFOA-S. Thus, the proposed BFOA-S is not inferior but competitive in term of testing error against the selected ten algorithms.

Table 5.12: The Statistical Result Comparison of Two Proposed Algorithms Against Ten Selected IS Algorithms using 24 Selected Datasets based on Testing Error Rate

Algorithm (Row)	**Analysis**	**Algorithm (Column)**	
		BFOA-S	BFOA-G
IB3	Ss	14/0/10	**18/0/6****
	Ps	0.541	0.023
	Pw	0.658	0.011
DROP3	Ss	12/0/12	17/0/7
	Ps	1.000	0.064
	Pw	0.290	0.081
ICF	Ss	12/0/12	**18/0/6****
	Ps	1.000	0.023
	Pw	0.458	0.019
GA-CHC	Ss	14/0/10	**18/0/6****
	Ps	0.541	0.023
	Pw	0.607	0.022
MSS	Ss	14/0/10	**18/0/6****
	Ps	0.541	0.023
	Pw	0.668	0.032
EbIS	Ss	14/0/10	17/0/7
	Ps	0.541	0.064
	Pw	0.819	0.059
IMOEA	Ss	15/0/9	**18/0/6****
	Ps	0.307	0.023
	Pw	0.648	0.005
LVQ-PRU	Ss	7/0/17	13/0/11
	Ps	0.064	0.839
	Pw	0.067	0.568
CCIS-10	Ss	7/0/17	16/0/8
	Ps	0.064	0.152
	Pw	0.028	0.424
IS-GA	Ss	**23/0/1****	**22/0/2****
	Ps	0.000	0.000
	Pw	0.000	0.000

Note: * the row algorithm has significant difference with the column algorithm for both statistical tests, **the column algorithm has significant difference with the row algorithm for both statistical tests

Table 5.13: The Statistical Result Comparison of Two Proposed Algorithms Against Ten Selected IS Algorithms Using 24 Selected Datasets Based on Storage Requirement

Algorithm (Row)	**Analysis**	**Algorithm (Column)**	
		BFOA-S	BFOA-G
IB3	Ss	**24/0/0****	**24/0/0****
	Ps	0.000	0.000
	Pw	0.000	0.000
DROP3	Ss	**24/0/0****	**24/0/0****
	Ps	0.000	0.000
	Pw	0.000	0.000
ICF	Ss	**24/0/0****	**24/0/0****
	Ps	0.000	0.000
	Pw	0.000	0.000
GA-CHC	Ss	**24/0/0****	**22/0/2****
	Ps	0.000	0.000
	Pw	0.000	0.000
MSS	Ss	**24/0/0****	**23/0/1****
	Ps	0.000	0.000
	Pw	0.000	0.000
EbIS	Ss	**24/0/0****	**23/0/1****
	Ps	0.000	0.000
	Pw	0.000	0.000
IMOEA	Ss	**19/0/5****	16/0/8
	Ps	0.007	0.152
	Pw	0.004	0.052
LVQ-PRU	Ss	**22/0/2****	**24/0/0****
	Ps	0.000	0.000
	Pw	0.000	0.000
CCIS-10	Ss	13/0/11	12/0/12
	Ps	0.839	1.000
	Pw	0.346	0.977
IS-GA	Ss	**19/0/5****	16/0/8
	Ps	0.007	0.152
	Pw	0.013	0.932

Note: *the row algorithm has significant difference with the column algorithm for both statistical tests, **the column algorithm has significant difference with the row algorithm for both statistical tests

The statistical results in Table 5.13 clearly shows that the proposed BFOA-G and BFOA-S have better performance in storage requirement compared to most of the selected algorithms. BFOA-G statistically outperformed seven algorithms and there is significant difference at 95% confidence level for both statistical tests. Meanwhile, the proposed BFOA-

S outperformed nine algorithms in term of storage requirement. There statistical result shows that there is significant difference in both statistical tests between BFOA-S and the nine algorithms. Both proposed algorithms have competitive performance against $CCIS_{10}$ since there is no significant difference in the statistical result.

Table 5.14: The Ranking of 12 Algorithms based on Average Testing Accuracy (Acc), Reduction Rate (Red) and Balancing Capability Between Accuracy and Reduction (Acc*Red)

	Avg. Testing Accuracy		**Reduction Rate**		**Acc*Red**
BFOA-G	**0.8310**	**BFOA-S**	**0.9625**	**BFOA-G**	**0.7937**
CCIS-10	0.8294	IS-GA	0.9566	CCIS-10	0.7776
LVQ-PRU	0.8277	**BFOA-G**	**0.9551**	**BFOA-S**	**0.7199**
DROP3	0.8156	CCIS-10	0.9375	IMOEA	0.7041
ICF	0.8066	IMOEA	0.9204	LVQ-PRU	0.6850
EbIS	0.7983	GA-CHC	0.8454	IS-GA	0.6752
GA-CHC	0.7979	IB3	0.8381	GA-CHC	0.6745
MSS	0.7968	LVQ-PRU	0.8276	IB3	0.6646
IB3	0.793	ICF	0.806	DROP3	0.6533
IMOEA	0.765	DROP3	0.801	ICF	0.6501
BFOA-S	**0.7479**	MSS	0.7048	MSS	0.5616
IS-GA	0.7058	EbIS	0.5356	EbIS	0.4276

On the other hand, a ranking for the proposed algorithms and ten selected algorithms is generated in the Table 5.14 according to their average testing accuracy (Acc), reduction rate (Red) and balancing capability between accuracy and reduction rate (Acc*Red). This additional analysis is generated to evaluate the capability of an algorithm to balance between classification accuracy and reduction for the purpose of comparison between the proposed algorithm and the other ten selected algorithms. The results were extracted from Table 5.10 for all algorithms. The ranking starts from the highest performance algorithm and ends with poorest performance algorithm at the bottom in every respective column.

Based on the result in table 5.14, the proposed BFOA-G is at top of ranking that indicates the highest performance algorithm for the average testing accuracy for 24 datasets. Unfortunately, the proposed BFOA-S is ranked as second from the bottom which is only outperformed IS-GA in term of average testing accuracy. However, BFOA-S has the highest performance in reduction rate and placed first in the ranking. BFOA-G also showed a good performance in reduction rate where it ranked as the third place in the ranking.

Since Instance Selection method aims for algorithm that has high performance in both accuracy and reduction rate then the products of testing accuracy and reduction rate is calculated to measure the capability of the algorithm to balance the trade-off between these two criteria. The result in Table 5.14 shows that the proposed BFOA-G obtained the first place in the ranking. This indicates that BFOA-G have the highest capability to produce a high accuracy solution with high reduction rate. It means that BFOA-G is capable to consider the trade-off between those criteria and able to excel in both of them. Apart from that, BFOA-S also shows a competitive result where it ranked as the third place in the ranking for Acc*Red. BFOA-S ranked as third as it loses the balance in considering the accuracy of the reference set and focussed more on reduction. Thus, it unable to outperformed $CCIS_{10}$ since this algorithm is consistent in term of testing accuracy and also have high reduction rate. It is expected that BFOA-S has this kind of result since BFOA-S adopted most of the conventional methods from IS-GA. Therefore, BFOA-S also inherit all the challenges or weaknesses faced by IS-GA. However, BFOA-S significantly outperformed IS-GA as it ranked in top three in the ranking and outperformed other algorithms in term of balancing capability of between reduction and accuracy (Acc*Red). Thus, BFOA-S is competitive classification algorithm against the recent IS algorithms despite of having most of the conventional methods in its implementation.

In addition, Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6 show scatter plots of the results for 24 datasets obtained from Table 5.10 for each algorithm. Each algorithm has two scatter plots which are for small datasets (<2000 instances) and medium datasets (>2000 instances). Similar with the previous section, *x*-axis represent the storage requirement and *y*-axis represent the testing error of the algorithm. The best result is determined by the algorithm with the most circles closer to origin (0,0).

Based on scatter plots for small datasets in Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6, there are roughly seven algorithms have been identified as they displayed better results in term of testing error for small and medium datasets. The best result was portrayed by $CCIS_{10}$ for small datasets with error below than 0.4 (< 0.4). BFOA-G and LVQPRU also shows competitive results in small datasets that mostly of the circles are positioned below 0.3 on *y*-axis (< 0.3) for the testing error which as good as the CCIS's performance. In term of storage requirement, $CCIS_{10}$ showed a better result where all circles are positioned below 0.1 (< 0.1). However, BFOA-G, BFOA-S and IS-GA also show competitive result in storage requirement for small datasets where most of the circles are positioned below 0.1 (< 0.1) and except have an outlier positioned more than 0.1 (< 0.2).

For medium datasets, LVQPRU and BFOA-G displayed better result compared other algorithms in term of testing error. All circles are positioned below 0.2 (< 0.2) in the scatter plot for LVQPRU. Meanwhile, the circles in scatter plot of BFOA-G for medium datasets are positioned below 0.3 (< 0.3). LVQPRU has better result compared to BFOA-G in medium datasets for testing error. In term of storage requirement, BFOA-G, BFOA-S, LVQPRU and IS-GA showed better result where most of the circles are positioned below than 0.1 (< 0.1) on *x*-axis for medium datasets. Among of them, BFOA-G has the lowest

storage requirement for medium datasets where the circles are closer to origin compared to others. As the summary for Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6, some algorithm performs better in small datasets and some is better in medium datasets. However, the proposed BFOA-G is competitive in both small and medium dataset in term of testing error and storage requirement.

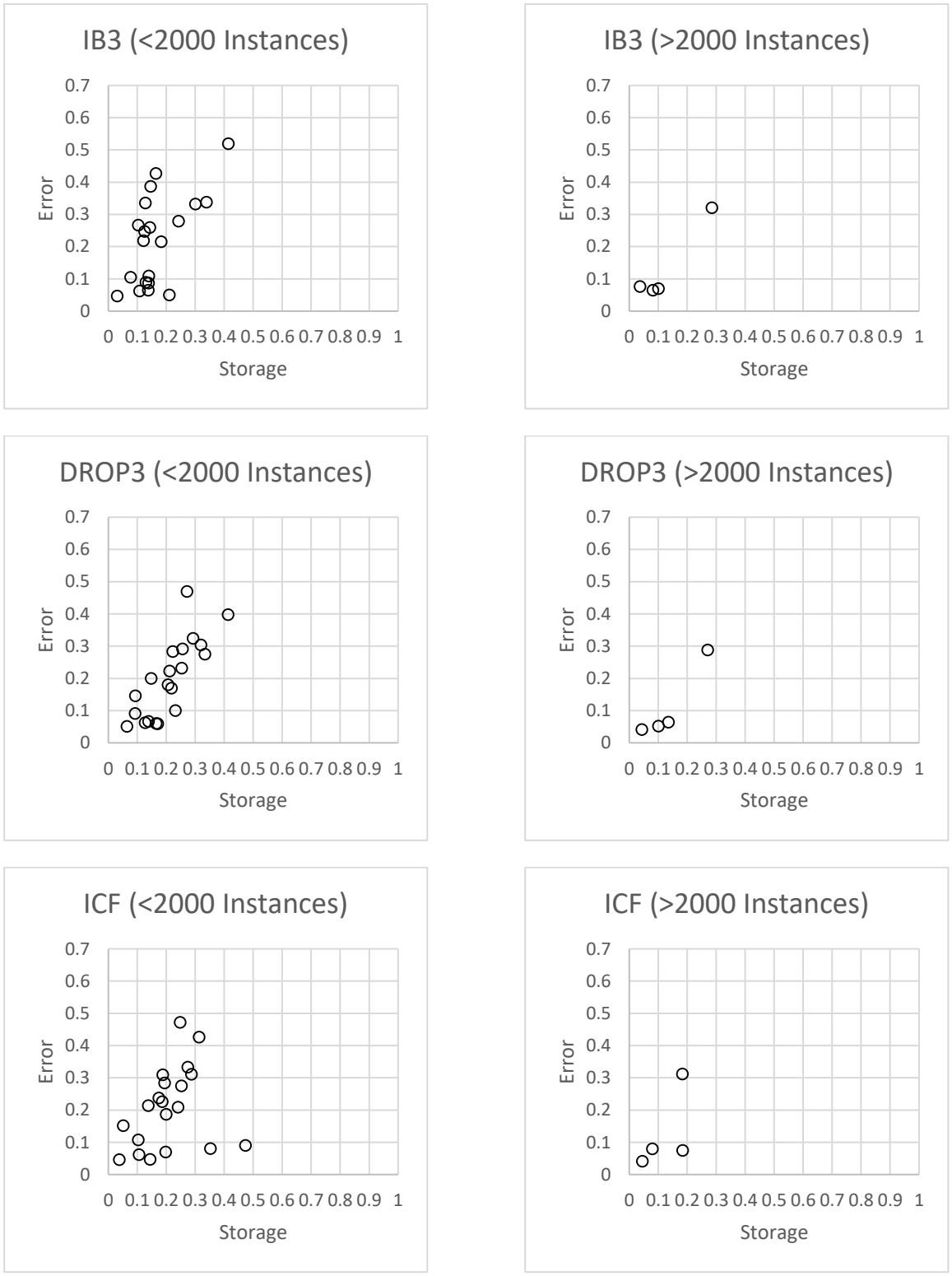


Figure 5.3: Summary of Results on Small and Medium Datasets for 12 Algorithm based on Testing Error Rate and Storage Requirement (Part 1)

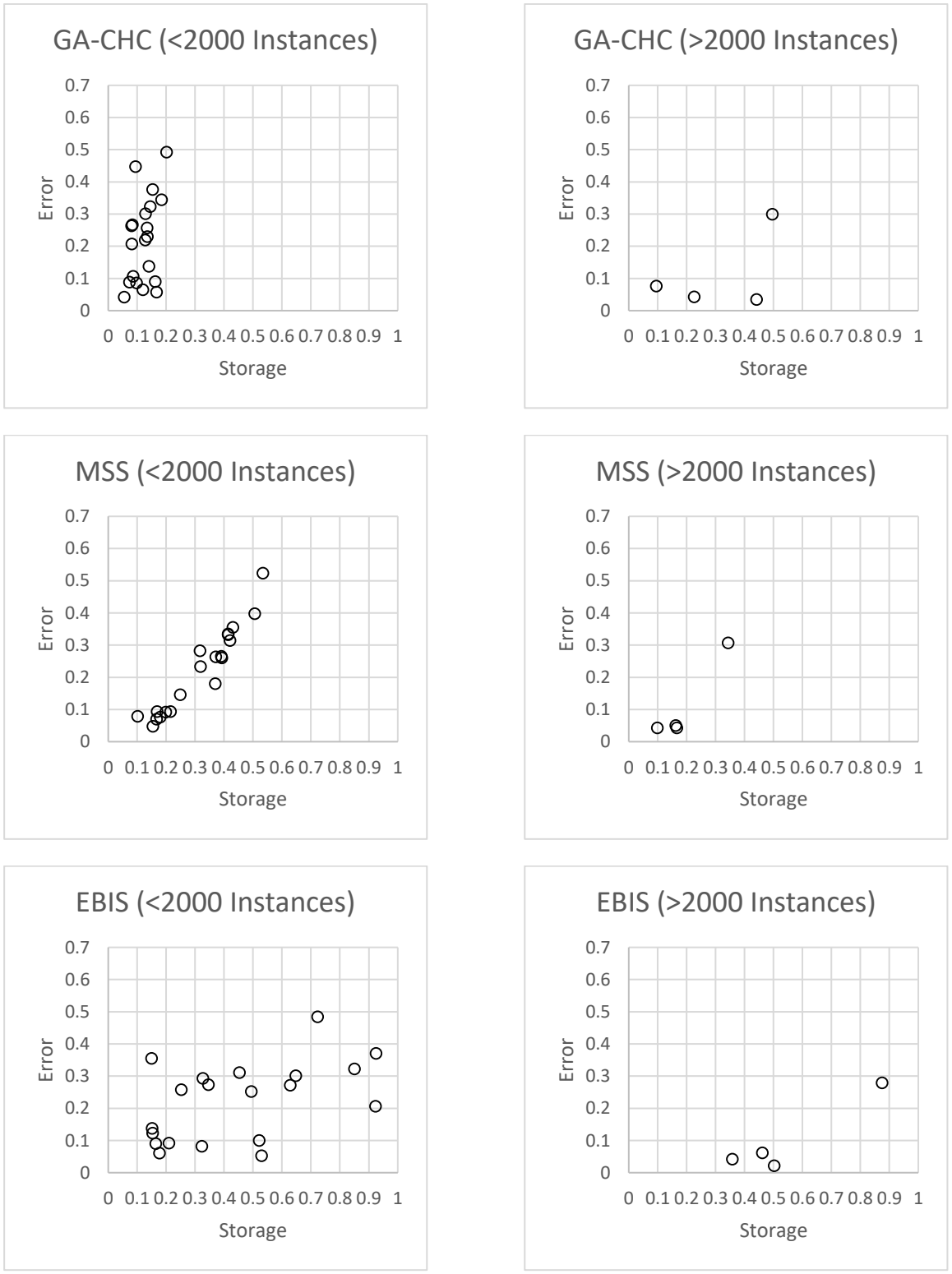


Figure 5.4: Summary of Results on Small and Medium Datasets for 12 Algorithm based on Testing Error Rate and Storage Requirement (Part 2)

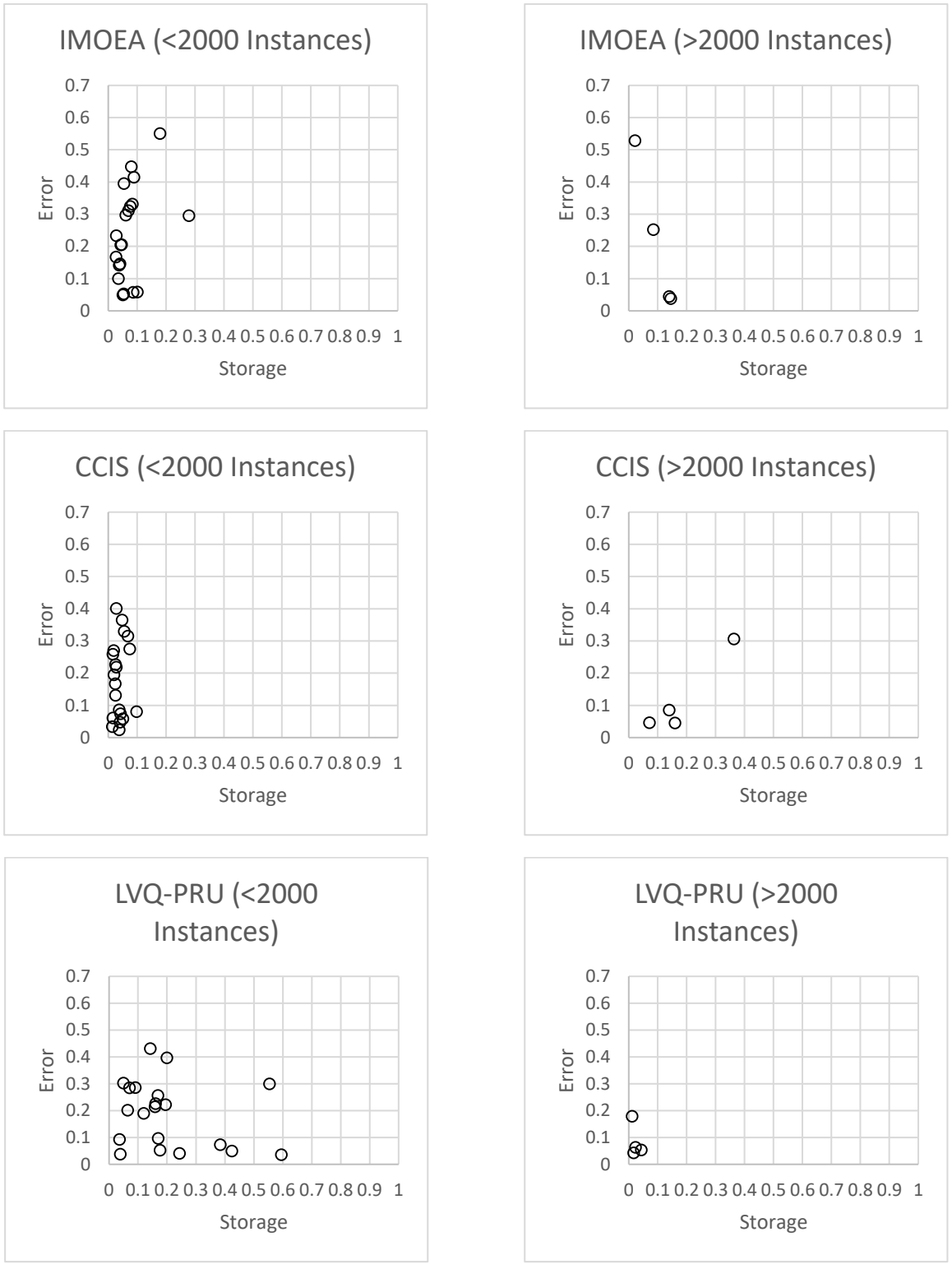


Figure 5.5: Summary of Results on Small and Medium Datasets for 12 Algorithm based on Testing Error Rate and Storage Requirement (Part 3)

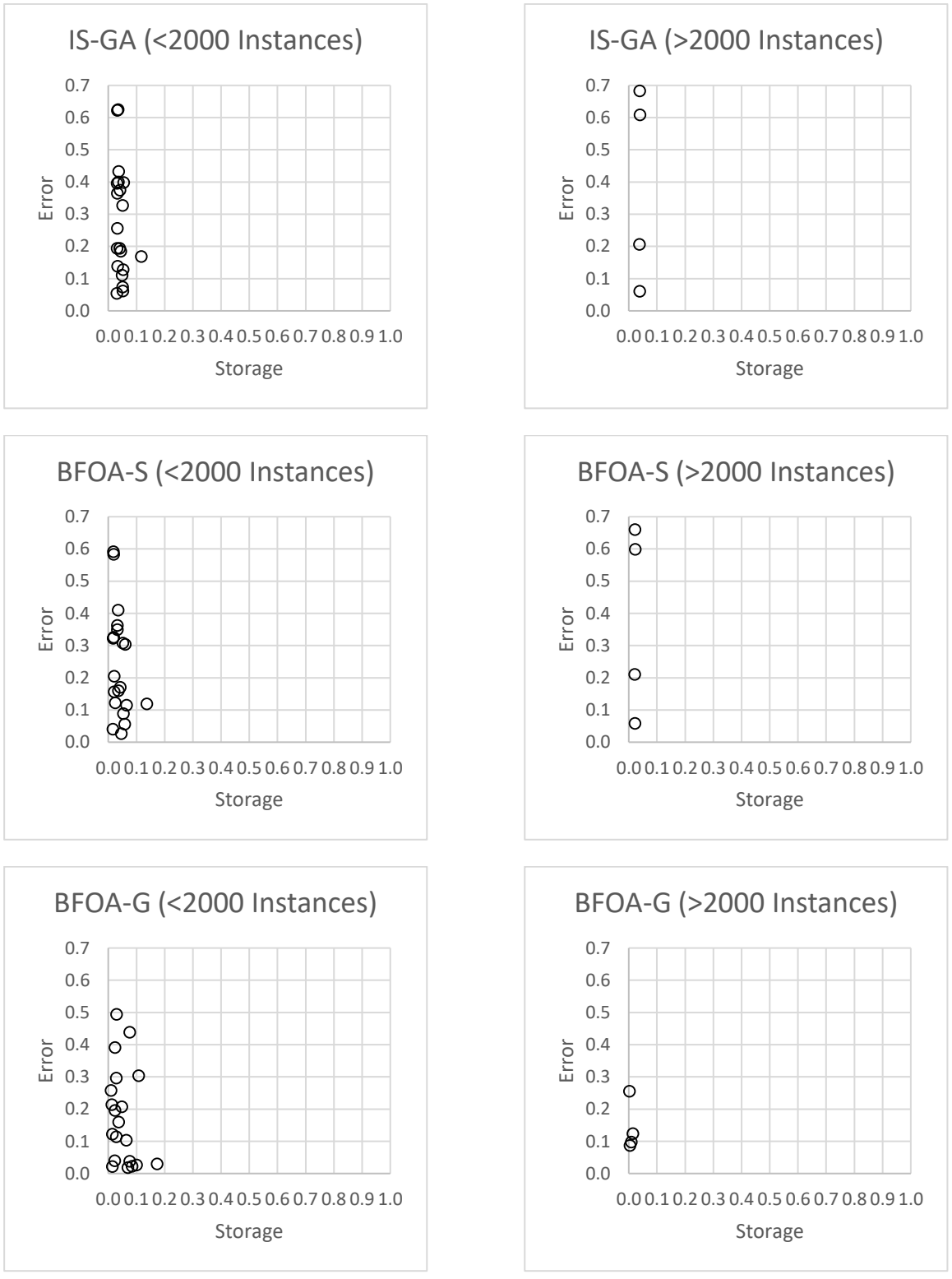


Figure 5.6: Summary of Results on Small and Medium Datasets for 12 Algorithm based on Testing Error Rate and Storage Requirement (Part 4)

5.6 Discussion

This study has successfully introduced and implemented two new conceptual frameworks for the conversion of the optimisation BFOA into Instance Selection (IS) algorithm that covers two major categories in Instance Selection which are Prototype Selection and Prototype Generation. The proposed BFOA for Prototype Selection is called as BFOA-S and the proposed BFOA for Prototype Generation is named as BFOA-G. From the experimental study that has been conducted, both proposed algorithms are able to outperform the benchmark algorithm statistically. This study has proved that it is possible to adopt an optimisation BFOA into IS algorithm and it also proved that the capabilities of the nature-inspired algorithm affect the performance of the future IS algorithm that will be developed.

From the experiment, BFOA-G is statically proved as the most efficient algorithm against the benchmark and other selected IS algorithms. On the other hand, the other proposed algorithm BFOA-S also outperforms the benchmark algorithms and shows competitive result against the other selected IS algorithms. Despite of having similar solution representation, fitness function and initialisation method with the conventional IS-GA, BFOA-S able to perform faster and produce higher accuracy reference set compared to IS-GA. This result shows that the proposed BFOA has better global search capability and higher convergence rate compare to GA for IS problem. This finding is consistent with the performance of BFOA in other domains against GA (Chakrabarty et al., 2012; Zhang & Wu, 2012; Rupal & Kataria, 2014; Hongwei & Liwei, 2015; Lv et al., 2018).

Unfortunately, BFOA-S unable to outperform the other state-of-the-art IS algorithm and the recent IS algorithms due to the challenges and weaknesses from the conventional

methods. It is undeniable that the original capabilities of an optimisation algorithm affect the capability of the future developed IS algorithm. It also expected for the BFOA-S to have this kind of performance against them since the BFOA-S adopted the conventional methods to convert an optimisation BFOA into evolutionary IS algorithm. The advantage of adopting the conventional method is that it easier to study the capability of BFOA as IS algorithm and observe either the modified BFOA is inheriting its original capabilities from its original optimisation algorithm or not. Besides, the study can observe how far the capability of BFOA can improve the nearest neighbour classification instead of having the most conventional method as the base for constructing the reference set. Based on this finding, it is easier to identify the weaknesses that can be improved in the future.

Apart from that, both proposed algorithms are faster and more efficient compared to benchmark algorithm in the experiment. It is because of the both proposed algorithms have lower cycles than IS-GA. Commonly, the algorithm's cycles are calculated based on the iteration of outermost loop counters. As stated in the parameter configuration, IS-GA has 500 generation cycles that halt the algorithm when it reaches 500 generations of population. On the other hand, the outermost loop for the proposed BFOA-S and BFOA-G is the elimination and dispersal steps, N_{ed} and both algorithms have small iteration compared to IS-GA. The value of N_{ed} for BFOA-G is 10 (N_{ed} = 10) and the value of N_{ed} BFOA-S is 5 (N_{ed} = 5). It is impossible to produce a good result by having such a small generation cycle of population. According to Li et al. (2014) in their analysis suggested that the cycles are determined by evolution generation that based on reproduction steps, N_{re} for BFOA since the population of the bacteria are evolved during reproduction process. Hence, the generation cycles are the product of N_{re} and N_{ed} for BFOA. The generation cycles for BFOA-S is 25 cycles (N_{re} =5 , N_{ed} =5) and the generation cycles for BFOA-G is 200 cycles (N_{re}=20 , N_{ed}

=10). Since both proposed algorithms have smaller generation cycles, they require less time and resources to complete the training process compared to benchmark IS-GA.

In comparison between BFOA-S and BFOA-G, there is huge difference in term of time complexity. BFOA-G works 13 times faster than the BFOA-S. BFOA-S unable to perform faster due to the complexity in the solution representation, encoding method and calculation of the fitness. The major contribution to the complexity of the BFOA-S is the length of the bacterium 's cells. Since a bacterium in BFOA-S represent a candidate reference set, the bacterium inherits the total size of dataset supplied as its cell's length. The larger the size of the dataset, the longer the size of a bacterium. Furthermore, every time there is changes occurred on the binary values in the cell, it will trigger the algorithm to re-evaluate the bacterium fitness. The algorithm needs to extract the instance's attributes based on binary value in the cells to carry out the calculation for the fitness. This situation increases resources needed for calculation and increases the time taken for the algorithm to complete its cycles. This finding is consistent with Cano et al. (2005) and Garcia et al. (2008) that the length of chromosome affects the computational time of the algorithms. Meanwhile in BFOA-G, the bacterium's cell size is depended on the dimension of an instance's attributes. Increasing in number of instances in the dataset will not affect the length of a bacterium. Furthermore, any modification is being made directly on the bacterium's attributes which also represents an instance's attributes. The process is simpler and occurred faster than BFOA-S since the bacterium's length is shorter. Thus, BFOA-G can perform faster to complete its cycles. The improvement for BFOA-S can be made by using different solution representation and encoding method or by doing some modification on the algorithm's structure to reduce the complexity of the algorithm.

On the other hand, both proposed algorithms able to produce better results despite of having small generation cycles due to the global search capability in BFOA. Generally, an algorithm with small cycles is unable to produce good result because there are not enough iterations for the algorithm to search and converge at the global solution. Sometimes, it is possible for the algorithm trapped at the local solution. Global search capability facilitates the algorithm to discover and converge at the global solution faster and avoid the algorithm from trapped at local solution. This can be seen clearly in the performance of BFOA-S against IS-GA where the global search capability of BFOA helps the algorithm to search for high performance reference set with small cardinality faster than IS-GA. Besides, BFOA-G also has similar capability that it can converge at the global solution with small generation cycles.

Another effect of the global search capability in the proposed algorithm is the reduction performance of the proposed algorithm. The global search ability does not affect the reduction performance of BFOA-G because the size of solution is fixed with the predefined rules that has been set from the start. However, the reduction performance of BFOA-S and IS-GA is controlled by similar predefined threshold T=10 for penalty function. According to the results, BFOA-S has higher reduction rate in all datasets compared to IS-GA. The global search capability that reside in elimination and dispersal process helps the proposed BFOA-S to explore into new search space. Therefore, BFOA-S able to find the smallest size possible reference set that can represent the datasets compared to IS-GA.

The current value of threshold T adopted from Kuncheva and Bezdek (1998) is not suitable for datasets that have class labels more than threshold value. Some classes will be missed out from the reference set since the algorithm is forced to reduce the cardinality of

the reference set equal or less than the threshold. It is suggested to have a flexible value of *T* to ensure that at least each class has a single instance that can represent them in the reference set. On top of that, penalty function with small threshold value only works on smaller size of data. When it comes to a larger size of dataset, the initial penalty value become too high because higher penalty is given to the extra instances in the reference set that above the threshold. Then, the removal or reduction in a reference set will not have so much impact on the fitness value of the reference set since the penalty value of the reduced reference set will be too small. This condition makes the algorithm become less sensitive to the changes in the classification accuracy of the reference set and it affects the decision-making mechanism of the algorithm to select the best reference set in latter process. Moreover, it also affects the capability of the algorithm to balance the trade-off between generalization accuracy and reduction. Thus, the algorithm will focus more on reduction and disregard the generalization performance of the reference set that can been in the result produced by BFOA-S and IS-GA.

Even though this study does not focus specifically on the class distribution in datasets, most of the datasets used in the experiment are imbalanced. Imbalanced dataset is known with a lot of issues and challenges for its majority and minority classes that affect the classification accuracy of the algorithm (He & Garcia, 2009; Zang & Wang, 2013). Balanced datasets normally produce better result compared to imbalance datasets (Zang & Wang, 2013). The classification performance for imbalanced dataset is depended on the capability of the algorithm to produce a good reference set that is not biased towards majority classes. As can be seen from the results, both proposed algorithms are able to produce high accuracy results that on par with the recent algorithms especially BFOA-G. It is possible because BFOA-G is adopted with restricted initialisation method that consider the ratio of instances

for each class. This method ensures that at least an instance is presented to become the representative for the minority class. Thus, it can reduce the effect of bias in the reference set for the testing process.

5.7 Summary

As the summary, this chapter has presented and discussed the results and analysis obtained from the experimental studies to evaluate the performance of the proposed algorithms, BFOA-S and BFOA-G in term of generalization accuracy, storage requirement, time complexity and the balancing capability between reduction and generalization. This chapter also discussed in detail the effect of the parameters on the algorithm's performance based on the heuristic setting. From the heuristic result, this study has come up with recommended parameter configurations for both proposed algorithms. Furthermore, the finding in heuristic result can be the guidelines for the future research to improve the proposed algorithms.

There are two experiments that have been carried out which are against the benchmark algorithms using 42 UCI datasets and a comparison against three state-of-the-art and nine recent IS algorithms using 24 selected datasets. For the experiment against the benchmark algorithms, the proposed BFOA-G is statistically most efficient and has the highest performance in overall evaluation criteria. It performed 39 times faster than the benchmark IS-GA algorithm and approximately 3 times faster than BFOA-S. It also has a good balancing capability since it able to produce high reduction on the storage requirement while maintaining high generalization accuracy simultaneously. The proposed algorithm obtained 80.73% on the average testing accuracy and it has 10% higher than the performance of IS-GA that only achieved 70.7% on the average testing accuracy for 42 datasets.

The comparison study against three state-of-the-art and nine recent IS algorithms shows that the proposed algorithm BFOA-G statistically outperformed the other algorithms in term of average error and storage requirement in general. A ranking has been established which based on the balancing capability between reduction and generalization of the algorithms. BFOA-G has the highest performance and ranked as the first place in term balancing capability for the evaluated criteria. On the other hand, BFOA-S also performed competitively as it ranked the third place in the ranking. Generally, both proposed algorithms perform well in both experiments. On the other hand, BFOA-S also performed competitively against the existing IS algorithm as it ranked the third place in the ranking. Furthermore, the statistical result for BFOA-S also shows that the existing IS algorithm do not outperform the proposed BFOA-S. Hence, it can conclude that BFOA-S has competitive performance against them.

CHAPTER 6

CONCLUSION

6.1 Conclusion

As an overall conclusion, this study has introduced two new Instance Selection (IS) algorithms which originated from Bacterial Foraging Optimisation Algorithm (BFOA) to address the problem of Nearest Neighbour Classification. The two algorithms were developed based on each category in IS, indicated in their names respectively, BFOA-S (BFOA using the concept of Prototype Selection) and BFOA-G (BFOA using the concept of Prototype Generation). BFOA is a nature-inspired optimisation algorithm that is known for its successfulness in many applications and domains. It has a good optimisation performance due to its global search capability and high convergence rate. However, none of the works published on BFOA has proposed BFOA as a classification algorithm despite its good performance compared to other nature-inspired algorithms such as Genetic Algorithm and Particle Swarm Optimisation. The two latter mentioned algorithm both have been adopted into IS algorithms in which they are well known to have better performance compared to other IS algorithms. In other words, this study has successfully introduced two new frameworks for the conversion of the optimisation BFOA into IS algorithms.

The performances of the proposed algorithms have been evaluated on 42 UCI datasets (Frank & Asuncion, 2011) with the selected benchmark algorithms. The statistical results demonstrate that both BFOA-S and BFOA-G were outperformed the other algorithms in term of their generalization accuracy, storage requirement, time complexity and capability to balance between reduction and generalization performance. BFOA-G was identified as

the most efficient and high-performance algorithm in the evaluated criteria, followed by BFOA-S.

In addition, a comparative study against three state-of-the-art algorithms and six recent IS algorithms have been done in term of their error rate, storage requirement and capability to balance between reduction and generalization performance. A ranking has been generated based on the algorithms' performance. From the ranking, BFOA-G has been identified as the most efficient and high-performance algorithm. On the other hand, BFOA-S also performed competitively as it ranked the third place in the ranking. Generally, both proposed algorithms were performed well in both experiments.

The proposed algorithms were able to produce good results due to the global search capability and its high convergence rate from the BFOA mechanism. The proposed frameworks were successfully preserved those capabilities even though lots of modification have been made in converting the BFOA into IS algorithm. Furthermore, this study also proves that the original capability of the nature-inspired algorithm does affect the performance of its modified version as an IS algorithm.

Based on the finding, this study has proved that the BFOA can be modified using PG and PS approach to become an IS classifier. The overall finding also shows that BFOA with the PG approach has better performance in term of accuracy, cardinality and time complexity compared to BFOA with PS approach. Therefore, it is recommended to adopt BFOA with PG approach since BFOA-G is the best modified IS algorithm in the study. Otherwise, it is necessary to find different initialisation method and fitness function in order to produce a better BFOA with PS approach so that its performance is comparable to BFOA with PG.

The proposed algorithms still have many rooms for improvement and still far from their state-of-the-art. In this study, few guidelines for parameter configuration of the proposed algorithms have been provided. These guidelines can be referred to as a further improvement of both proposed algorithms.

6.2 Future Works

Based on the evaluation and analysis obtained in this study, there are several issues that could be proposed for the future works. The issues are listed as follows.

Initialisation and representation method- As the experimental result in Chapter 5 suggests, new solution representation and initialisation method are needed to improve the performance of BFOA-S. The current binary-like representation of dataset in the proposed BFOA-S increases the complexity of the classifier to process and calculate the fitness function of candidate solutions since the length of chromosome increase proportionally with the size of dataset. Due to the complexity of the classifier, it is impossible to increase the iteration or adopt more complex method in latter process to improve BFOA-S. Thus, new initialisation and representation method is needed to improve the performance of BFOA-S.

Adopt or devise a better evaluation fitness function – To improve the classification accuracy for imbalanced datasets, a better performance measure needs to be adopted or devised. The current fitness function is measured solely based on the overall recognition and it does not consider the performance of data in each class. The weakness of the accuracy as the fitness function is the algorithm will prioritize the majority class since it will increase the overall recognition and might disregard the minority class to be selected as representative in the reference set or model. Thus, a better fitness function needs to be adopted to address the weakness.

Feature selection/filtering – In the literature, feature selection is used to improve the classification accuracy of the classifier by selecting the optimal features and removing the noises features. By removing noise features, it improves the accuracy of the representative to represent a respective class and simultaneously reduce the time complexity of classifier since the dimension of the dataset is reduced. This method can be adopted into BFOA-S and BFOA-G to improve their performance in classification.

Large datasets – This study does not include large datasets in the experiments due to scaling up problem and limited hardware resources. Most of evolutionary IS algorithms suffered scaling up problem when the size of dataset is too large to be processed. The current structures of the proposed algorithms were restricted to work with large dataset. Thus, the proposed algorithms need to be reconstructed or adopted with another methods for large dataset such as stratification method (García et al., 2012) to enable the proposed algorithms to work with large datasets (more than 20000 instances).

Pre-processing Algorithm – As mentioned in the discussion, Instance Selection algorithms also can be used as a pre-processing algorithm to find a good training set for other classifiers. Thus, the proposed BFOA can be used for pre-processing algorithm for other algorithms.

REFERENCES

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based Learning Algorithms. *Machine Learning*, *6*(1), 37-66.

Ahmad, S. S. S. (2014, December). Feature and Instances Selection for Nearest Neighbor Classification via Cooperative PSO. In *2014 4th World Congress on Information and Communication Technologies*, 45-50.

AL-Hadi, I. A. A., Hashim, S. Z. M., & Shamsuddin, S. M. H. (2011). Bacterial Foraging Optimization Algorithm for Neural Network Learning Enhancement. In *2011 11th International Conference on Hybrid Intelligent Systems*, 200-205.

Bensujin, B., Vijila, C. K., Hemanth, J., & Hubert, C. (2016). Frequency Dependent Adaptive Chemotaxis in Bacterial Foraging Optimization for ST Segment Elevation Myocardial Infarction Prediction. *Indian Journal of Science and Technology*, *9*(1), 1-9.

Bezdek, J. C., & Kuncheva, L. I. (2001). Nearest Prototype Classifiers Design: An Experimental Study. *International Journal of Intelligent Systems, 16*(12), 1445-1473.

Brighton, H., & Mellish, C. (2002). Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery*, *6*(2), 153-172.

Cano, J., Herrera, F., & Lozano, M. (2005). A Study on the Combination of Evolutionary Algorithms and Stratified Strategies for Training Set Selection in Data Mining. *Applied Soft Computings*, *6*(3), 323-332.

Cano, J. R., Herrera, F., & Lozano, M. (2003). Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD : An Experimental Study. *IEEE Transactions on Evolutionary Computation, 7*(6), 561-575.

Cano, J. R., Herrera, F., & Lozano, M. (2005). Stratification for Scaling up Evolutionary Prototype Selection. *Pattern Recognition Letters*, *26*(7), 953-963.

Carter, J. H. (2000). The Immune System as a Model for Pattern Recognition and Classification. *Journal of the American Medical Informatics Association*, *7*(1), 28-41.

Cervantes, A., Galván, I., & Isasi, P. (2007). An adaptive Michigan Approach PSO for Nearest Prototype Classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 287-296.

Cervantes, A., Galvain, I., & Isasi, P. (2005). A Comparison between the Pittsburgh and Michigan Approaches for the Binary PSO Algorithm. *2005 IEEE Congress on Evolutionary Computation*, *1*, 290-297.

Cervantes, A., Galván, I. M., & Isasi, P. (2009). AMPSO : A New Particle Swarm Method for Nearest Neighborhood Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(5), 1082-1091.

Chakrabarty, A., Choudhury, O., Sarkar, P., Paul, A., & Sarkar, D. (2012). Hyperspectral Image Classification Incorporating Bacterial Foraging-optimized Spectral Weighting, *Artificial Intelligent Research*, *1*(1), 63-83.

Chen, C. H., Su, M. T., Lin, C. J., & Lin, C. T. (2014). A Hybrid of Bacterial Foraging Optimization and Particle Swarm Optimization for Evolutionary Neural Fuzzy Classifier. *International Journal of Fuzzy Systems*, *16*(3), 422-433.

Cho, J. H., & Kim, D. H. (2011). Intelligent Feature Selection by Bacterial Foraging Algorithm and Information Theory. In *International Conference on Advanced Communication and Networking*, 238-244.

Chuang, L. Y., Tsai, S. W., & Yang, C. H. (2011). Catfish Binary Particle Swarm Optimization for Feature Selection. In *International Conference on Machine Learning and Computing IPCSIT*, *3*, 40-44.

Damodaram, R., & Phil, M. (2013). Bacterial Foraging Optimization for Fake Website Detection. *International Journal of Computer Science & Applications*, *1*(11), 116-127.

Darvishi, M., Javadi, H. H. S., Pedram, M. M., & Erfani, H. (2012). Using Regression in Bacterial Foraging Algorithm to Improve Robots Routing. *Research Journal of Applied Sciences, Engineering and Technology*, *4*(20), 405-4054.

Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. *Foundations of Computational Intelligence*, *3*, 23-55.

Decaestecker, C. (1997). Finding Prototypes for Nearest Neighbour Classification by Means of Gradient Descent and Deterministic Annealing. *Pattern Recognition, 30*(2), 281-288

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, *7*, 1-30.

Derrac, J., García, S., & Herrera, F. (2010). A Survey on Evolutionary Instance Selection and Generation. *International Journal of Applied Metaheuristic Computing*, *1*(1), 60-92.

Feng, H., Horng, J.-H., & Jou, S.-M. (2012). Algorithm Based Fuzzy-VQ Compression Systems. *Journal of Information Hiding and Multimedia Signal Processing*, *3*(3), 227-239.

Figueredo, G., Ebecken, N., Augusto, D., & Barbosa, H. (2012). An Immune-Inspired Instance Selection Mechanism for Supervised Classification. *Memetic Computing*, *4*(2), 135-147.

Fister Jr, I., Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). A Brief Review of Nature-Inspired Algorithms for Optimization. *Electrotechnical Review*, *80*(3), 1-7.

Frank, A., & Asuncion, A. (2011). *UCI Machine Learning Repository Irvine* [Data file]. Retrieved from http://archive.ics.uci.edu/ml/datasets.html

Garcia-Pedrajas, N., Del Castillo, J. A. R., & Ortiz-Boyer, D. (2010). A Cooperative Coevolutionary Algorithm for Instance Selection for Instance-based Learning. *Machine Learning*, *78*(3), 381-420.

Garcia, S., Derrac, J., Cano, J. R., & Herrera, F. (2012). Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *34*(3), 417-435.

García, S., Cano, J. R., & Herrera, F. (2008). A Memetic Algorithm for Evolutionary Prototype Selection: A Scaling Up Approach. *Pattern Recognition*, *41*(8), 2693-2709.

Geva, S., & Sitte, J. (1991). Adaptive Nearest Neighbor Pattern Classification. *IEEE Transactions on Neural Networks*, *2*(2), 318-322.

Gil-Pita, R., & Yao, X. (2007). Using A Genetic Algorithm for Editing K-Nearest Neighbor Classifiers. In *International Conference on Intelligent Data Engineering and Automated Learning*, 1141-1150.

Hamamoto, Y., Uchimura, S., & Tomita, S. (1997). A Bootstrap Technique for Nearest Neighbor Classifier Design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(1), 73-79.

He, H., & Garcia, E. a. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263-1284.

Ho, S. Y., Liu, C. C., & Liu, S. (2002). Design of an Optimal Nearest Neighbor Classifier Using an Intelligent Genetic Algorithm. *Pattern Recognition Letters*, *23*(13), 1495-1503.

Hossin, M. H., Mahudin, F., Din, I. J., & Mat, A. R. (2017). Analysis of Nine Instance-Based Genetic Algorithm Classifiers Using Small Datasets. *Journal of Telecommunication, Electronic and Computer Engineering*, *9*(3-11), 67-71.

Hongwei, Z., & Liwei, T. (2015). Cooperative Approaches to Bacterial Foraging Algorithm for Clustering. *International Journal of Database Theory and Application*, *8*(4), 81–90.

Hu, W., & Tan, Y. (2015). Prototype Generation Using Multiobjective Particle Swarm Optimization for Nearest Neighbor Classification. *IEEE Transactions on Cybernetics*, *46*(12), 2719-2731.

Ishibuchi, H., Nakashima, T., & Murata, T. (1997). Comparison of the Michigan and Pittburgh Approaches to Design of fuzzy classification system. *Electronics and Communications in Japan*, *80*(2), 379-387.

Jakhar, R., Kaur, N., & Singh, R. (2011). Face Recognition Using Bacteria Foraging Optimization-Based Selected Features. *International Journal of Advanced Computer Science and Applications*, *1*(3), 106-111.

Kaur, R., & Kaur, B. (2014). Artificial Neural Network Learning Enhancement Using Bacterial Foraging Optimization Algorithm. *International Journal of Computer Applications*, *102*(10), 27-33.

Kim, D. H., & Cho, J. H. (2005). Adaptive Tuning of PID Controller for Multivariable System Using Bacterial Foraging Based Optimization. In *International Atlantic Web Intelligence Conference*, 231-235.

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, *78*(9): 1464–1480.

Koplowitz, J., & Brown, T. A. (1981). On the Relation of Performance to Editing in Nearest Neighbor Rules. *Pattern Recognition*, *13*(3), 251-255.

Kora, P., & Kalva, S. R. (2015). Hybrid Bacterial Foraging and Particle Swarm Optimization for detecting Bundle Branch Block. *Springer Plus*, *4*(1), 1-19.

Kruatrachue, B., & Hongsamart, M. (2008). Prototype Selection based on Minimal Consistent Subset and Genetic Algorithms. In *SICE Annual Conference*, 682-686.

Kumar, D. P. (2014). Ground Water Prediction Using Bacterial Foraging Optimization Technique. *American Journal of Computing Research Repository*, *2*(3), 44-48.

Kuncheva, L. I. (1995). Editing for the k-Nearest Neighbors Rule by a Genetic Algorithm. *Pattern Recognition Letters*, *16*(8), 809-814.

Kuncheva, L. I. (1997). Fitness functions in Editing k-NN Reference Set by Genetic Algorithms. *Pattern Recognition*, *30*(6), 1041-1049.

Kuncheva, L. I., & Bezdek, J. C. (1998). Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search? *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *28*(1), 160-164.

Lei, X., Wu, S., Ge, L., & Zhang, A. (2011). Clustering PPI Data Based on Bacteria Foraging Optimization Algorithm. In *2011 IEEE International Conference on Bioinformatics and Biomedicine*, 96-99.

Li, J. (2014). Analysis and Improvement of the Bacterial Foraging Optimization Algorithm. *Journal of Computing Science and Engineering*, *8*(1), 1-10.

Li, J., Manry, M. T., Yu, C., & Wilson, D. R. (2005). Prototype Classifier Design with Pruning. *International Journal on Artificial Intelligence Tools*, *14*(1–2), 261–280.

Liu, H., & Motoda, H. (2002). On Issues of Instance Selection. *Data Mining and Knowledge Discovery*, *6*(2), 115-130.

Lozano, M., Sotoca, J. M., Sánchez, J. S., Pla, F., Pekalska, E., & Duin, R. P. W. (2006). Experimental Study on Prototype Optimization Algorithms for Prototype-Based Classification in Vector Spaces. *Pattern Recognition*, *39*(10), 1827-1838.

Lv, X., Chen, H., Zhang, Q., Li, X., Huang, H., & Wang, G. (2018). An Improved Bacterial-Foraging Optimization-Based Machine Learning Framework for Predicting The Severity of Somatization Disorder. *Algorithms*, *11*(2), 1-18.

Miloud-Aouidate, A., & Baba-Ali, A. R. (2013). An Efficient Ant Colony Instance Selection Algorithm for KNN Classification. *International Journal of Applied Metaheuristic Computing*, *4*(3), 47-64.

Miloud-Aouidate, A., & Baba-Ali, A. R. (2012). A Hybrid KNN-Ant Colony Optimization Algorithm For Prototype Selection. In *International Conference on Neural Information Processing*, 307-314.

Mitra, S., Pal, S. K., & Mitra, P. (2002). Data mining in Soft Computing Framework: A Survey. *IEEE Transactions on Neural Networks*, *13*(1), 3-14.

Nanni, L., & Lumini, A. (2009). Particle swarm optimization for prototype reduction. *Neurocomputing*, *72*(4–6), 1092-1097.

Nanni, L., & Lumini, A. (2011). Prototype Reduction Techniques: A Comparison Among Different Approaches. *Expert Systems with Applications*, *38*(9), 11820-11828.

Olesen, J., Cordero H, J., & Zeng, Y. (2009). Auto-Clustering Using Particle Swarm Optimization and Bacterial Foraging. In *International Workshop on Agents and Data Mining Interaction*, 69-83.

Pal, M., Bhattacharyya, S., Roy, S., Konar, A., Tibarewala, D. N., & Janarthanan, R. (2014). A Bacterial Foraging Optimization and Learning Automata Based Feature Selection for Motor Imagery EEG Classification. In *2014 International Conference on Signal Processing and Communications*, 1-5.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). Data Mining with an Ant Colony Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, *6*(4), 321-332.

Passino, K. M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control. *IEEE Control Systems Magazine*, *22*(3), 52-67.

Putra, I. M. B. P., Hartati, R. S., Putra, I. K. G. D., & Wirdiani, N. K. A. (2014). Optimized Back Propagation Learning in Neural Networks with Bacterial Foraging Optimization to Predict Forex Gold Index (XAUUSD). *Applied Mathematical Sciences*, *8*(38), 1847-1854.

Rani, B., Goel, A. K., & Kaur, R. (2016). A Modified Approach for Lung Cancer Detection Using Bacterial Forging Optimization Algorithm. *International Journal of Scientific Research Engineering & Technology*, *5*(1), 39-42.

Rani, D., & Mangat, V. (2013). An Efficient Technique for Disease Diagnosis Using Bacterial Foraging Optimization and Artificial Neural Network. In *2013 International Symposium on Computational and Business Intelligence*, 100-104.

Rosales-Pérez, A., Escalante, H. J., Coello, C. A. C., Gonzalez, J. A., & Reyes-Garcia, C. A. (2014, July). An Evolutionary Multi-Objective Approach for Prototype Generation. In *2014 IEEE Congress on Evolutionary Computation*, 1100-1107.

Rosales-Pérez, A., Gonzalez, J. A., Coello, C. A. C., Reyes-Garcia, C. A., & Escalante, H. J. (2016). EMOPG+ FS: Evolutionary Multi-Objective Prototype Generation and Feature Selection. *Intelligent Data Analysis*, *20*(1), 37-51.

Al-Shalabi, L. (2011). Knowledge Discovery Process: Guide Lines for New Researchers. *Journal of Artificial Intelligence*, *4*(1), 21-28.

Shivakumar, B. L., & Amudha, T. (2012). A Hybrid Bacterial Foraging Algorithm For Solving Job Shop Scheduling Problems. *Global Journal of Computer Science and Technology*, *12*(10), 1-11.

Sindhu, V. (2016). An Efficient Technique to Cancer Classification Using Fast Improved Bacterial Foraging. *International Journal of Recent Trends in Engineering & Research*, *2*(4), 436-441.

Skalak, D. B. (1994). Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In *Machine Learning Proceedings 1994*, 293-301.

Song, Q., Guo, X., & Li, H. (2016). Bacterial Foraging Optimization Based on LS-SVM for BTP Forecasting. *International Journal of Hybrid Information Technology*, *9*(1), 387-396.

Song, Q., & Wang, A.-M. (2013). A Hybrid Algorithm Based on LS-SVM and Bacterial Foraging Approach for Burning-Through-Points (BTP) Prediction in Sintering Process. *Journal of Theoretical & Applied Information Technology*, *49*(1), 231-240.

Suguna, N., & Thanushkodi, K. (2010). An Improved K-Nearest Neighbor Classification Using Genetic Algorithm. *International Journal of Computer Science Issues*, *7*(2), 18-21.

Szabo, A., & de Castro, L. N. (2013). A Constructive Data Classification Version of the Particle Swarm Optimization Algorithm. *Mathematical Problems in Engineering 2013,* 1-13.

Triguero, I., Derrac, J., Garcıa, S., & Herrera, F. (2011). Prototype Generation for Nearest Neighbor Classification: Survey of Methods. *Technical Report, Department of Computer Science and Artificial Intelligence,* University of Granada, Spain.

Triguero, I., Derrac, J., García, S., & Herrera, F. (2012). A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *42*(1), 86-100.

Triguero, I., García, S., & Herrera, F. (2011). Differential Evolution for Optimizing the Positioning of Prototypes in Nearest Neighbor Classification. *Pattern Recognition*, *44*(4), 901-916.

Tripathy, M., Mishra, S., Lai, L. L., & Zhang, Q. P. (2006). Transmission Loss Reduction Based On FACTS and Bacteria Foraging Algorithm. *Parallel Problem Solving from Nature-PPSN IX*, 222-231.

Varghese, T., Kumari, R. S., Mathuranath, P. S., & Singh, N. A. (2012). Performance Evaluation of Bacterial Foraging Optimization Algorithm for the Early Diagnosis and Tracking of Alzheimer's Disease. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, 41-48.

Wan, M., Li, L., Xiao, J., Wang, C., & Yang, Y. (2012). Data Clustering Using Bacterial Foraging Optimization. *Journal of Intelligent Information Systems*, *38*(2), 321–341.

Wilson, D. R., & Martinez, T. R. (2000). Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, *38*(3), 257-286.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., & Zhou, Z. H. (2008). Top 10 Algorithms in Data Mining. *Knowledge and Information Systems*, *14*(1), 1-37.

Xie, Q., Laszlo, C. A., & Ward, R. K. (1993). Vector Quantization Technique for Nonparametric Classifier Design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(12), 1326-1330.

Zhang, Y., & Wang, D. (2013). A Cost-Sensitive Ensemble Method for Class-Imbalanced Datasets. *Abstract and Applied Analysis*, *2013*, 1-6.

Zhang, Y., & Wu, L. (2012). Bacterial Foraging Optimization Used in Cluster Analysis. *International Journal of Digital Content Technology and Its Applications*, *6*(22), 345-354.

APPENDIX

LIST OF PUBLICATION

Published Articles

- Hossin, M., & Mohd Suria, F. (2017). A Conceptual Framework of Bacterial Foraging Optimization Algorithm for Data Classification. *Journal of Engineering and Applied Sciences*, *12*(Special issue 9), 8531-8536.

Exhibitions

1. Innovation Technology Expo 2016 (INTEX16), UNIMAS -Silver.
2. Innovation Technology Expo 2017(INTEX17), UNIMAS – Gold.
3. Malaysia Technology Expo 2017 (MTE 17), Kuala Lumpur – Bronze.
4. International Invention, Innovation & Technology Exhibition 2018 (ITEX'18), Kuala Lumpur – Bronze.

Intellectual Property

- A Method for Optimizing Instance Based Bacterial Foraging. (MYIPO IP NO: 2017700482) – Pending.